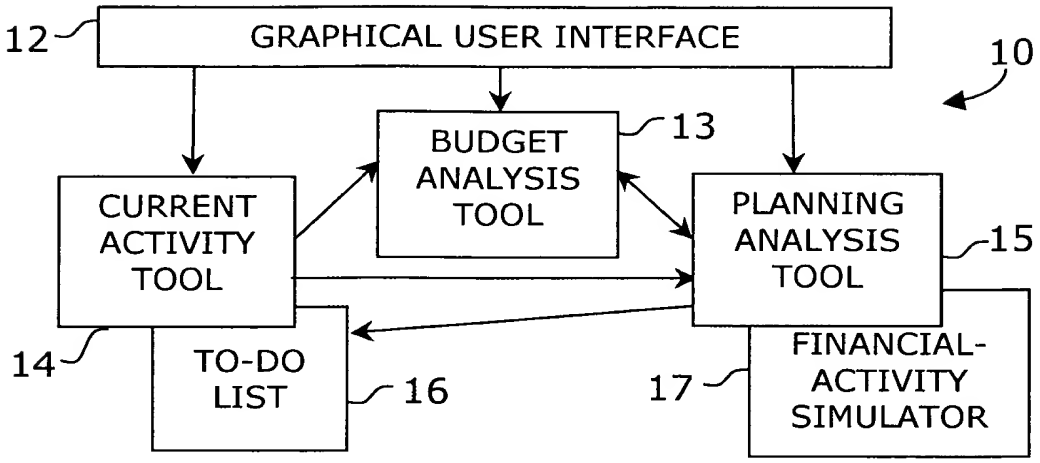


FIG. 1

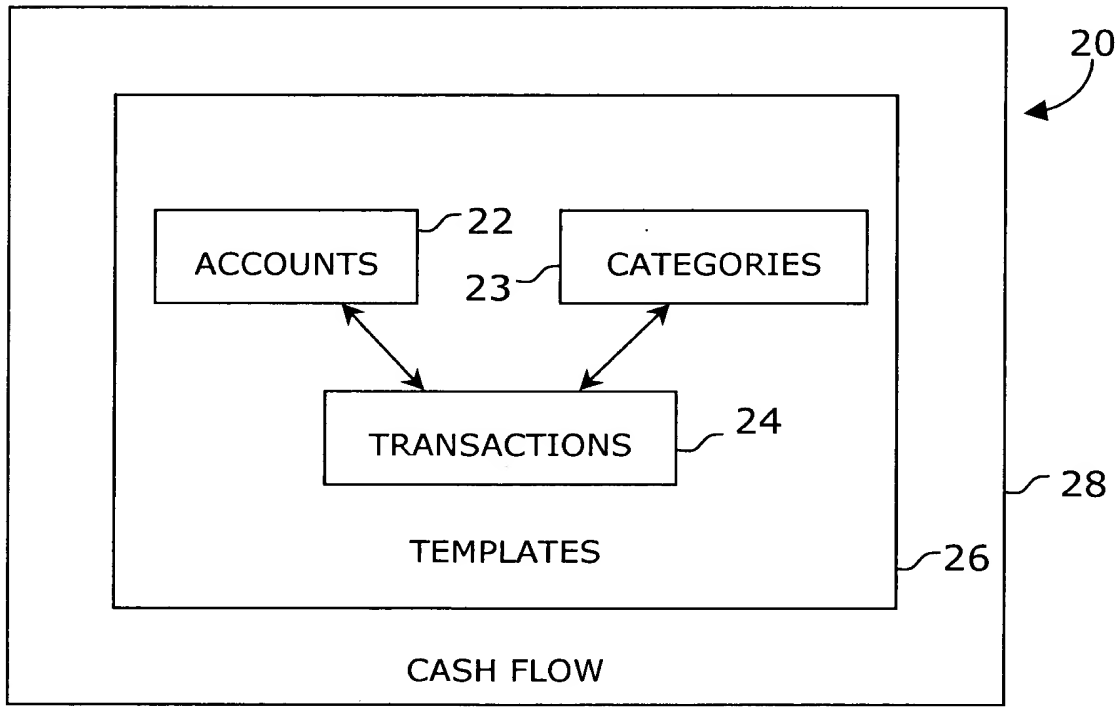


FIG. 2

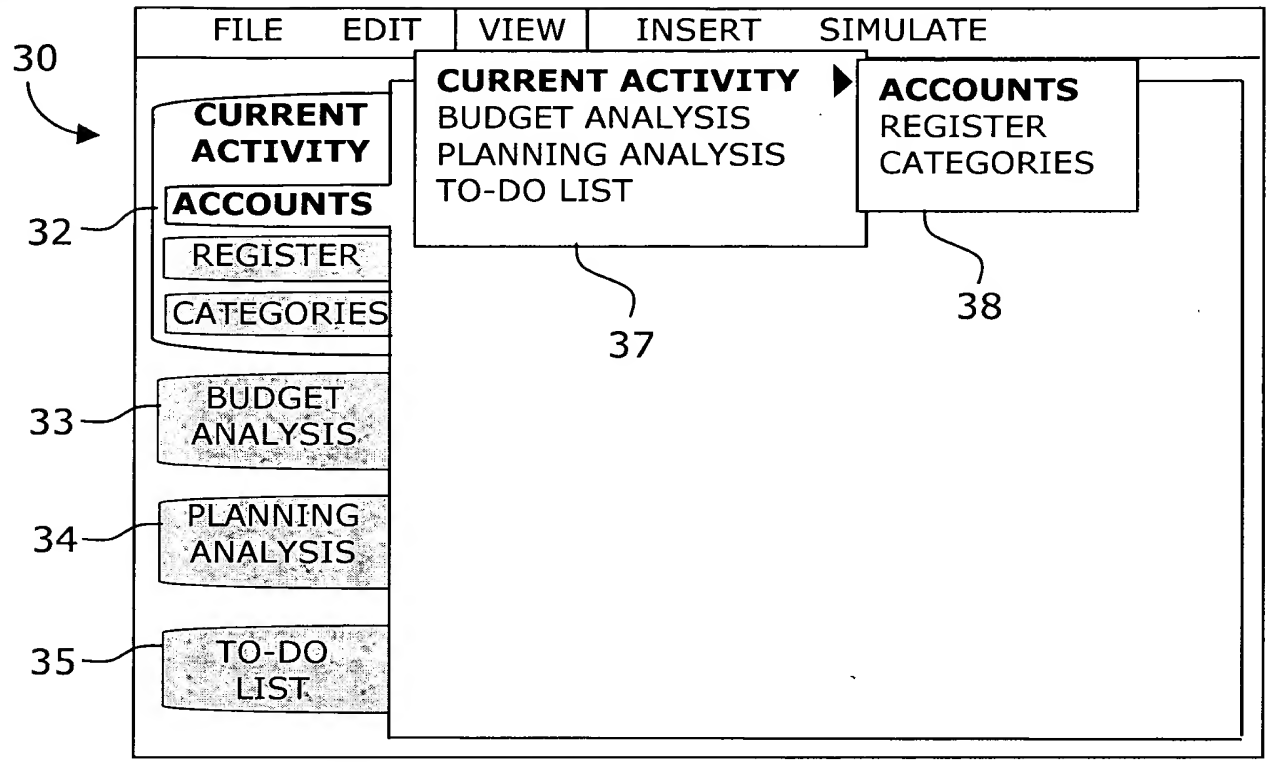


FIG. 3

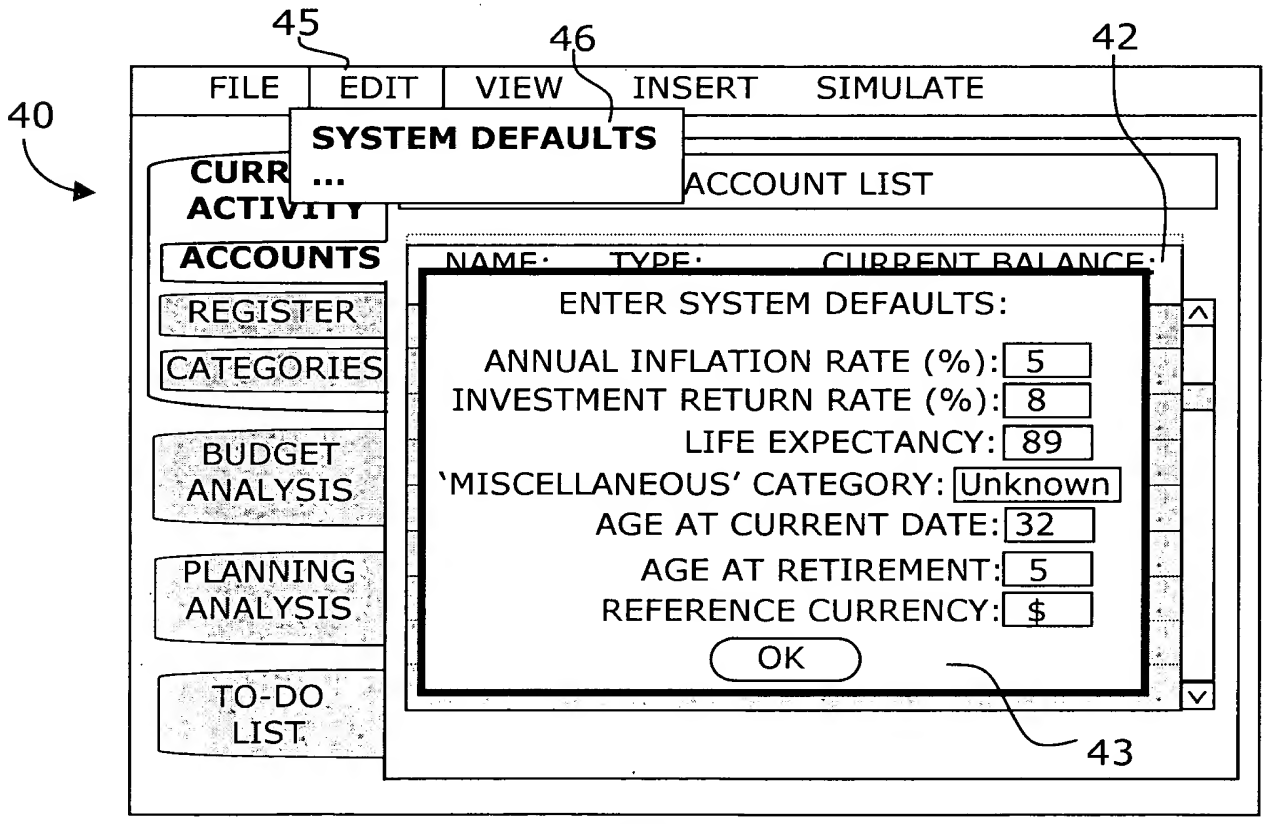


FIG. 4

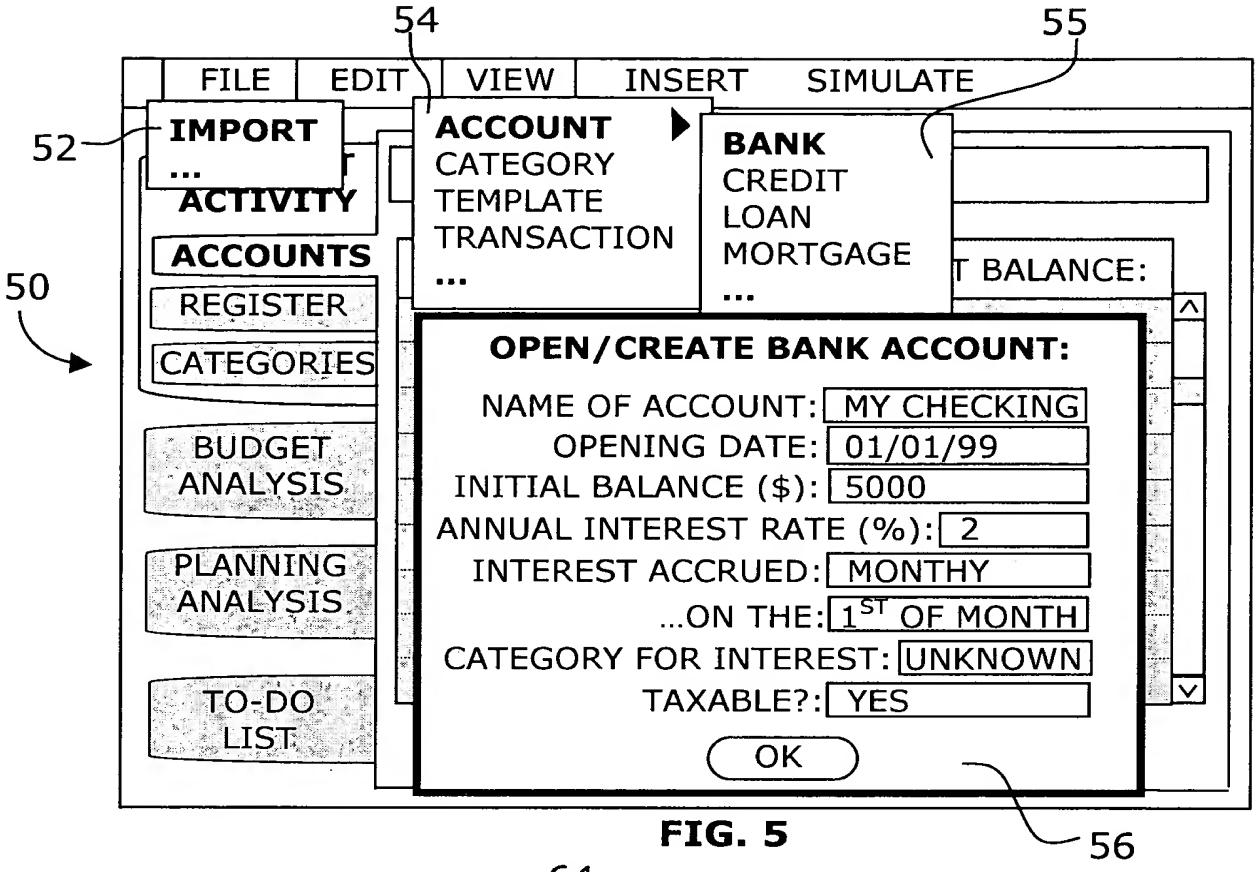


FIG. 5

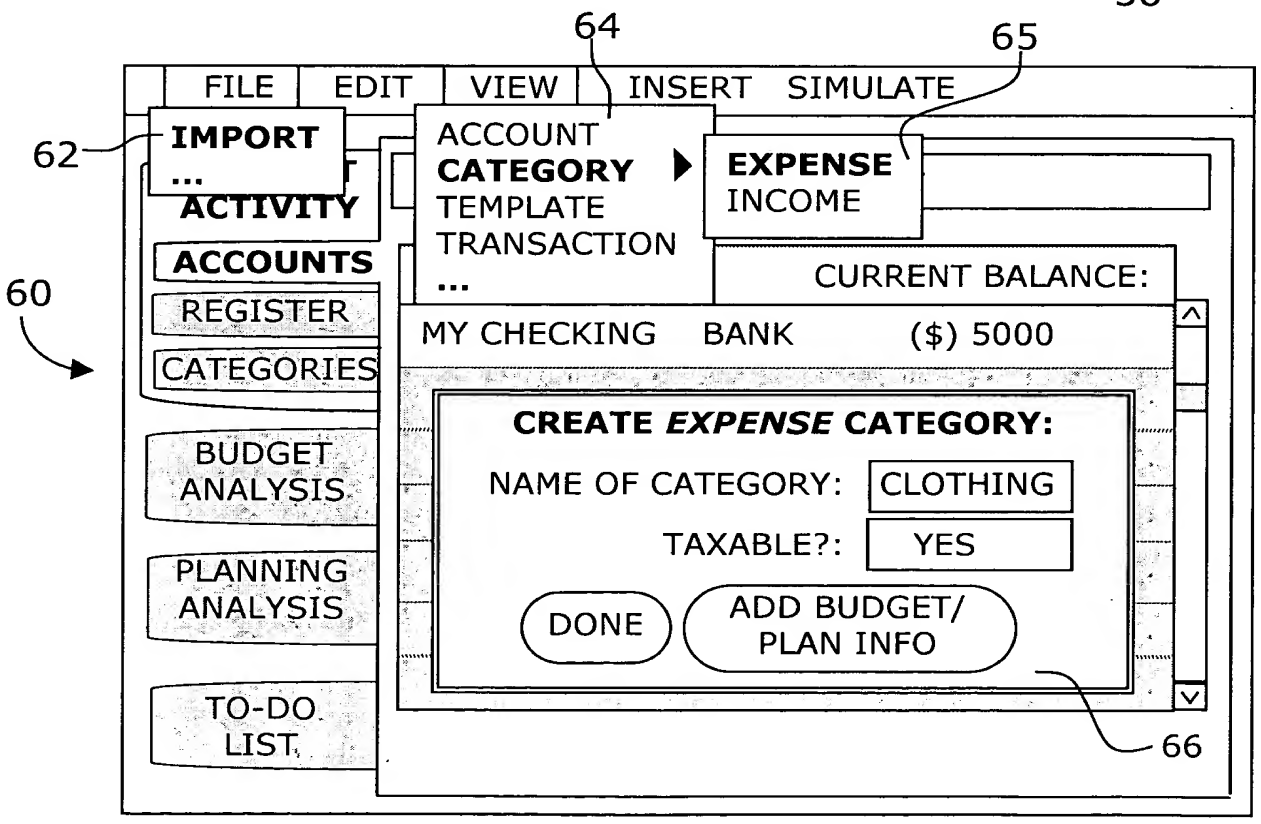


FIG. 6

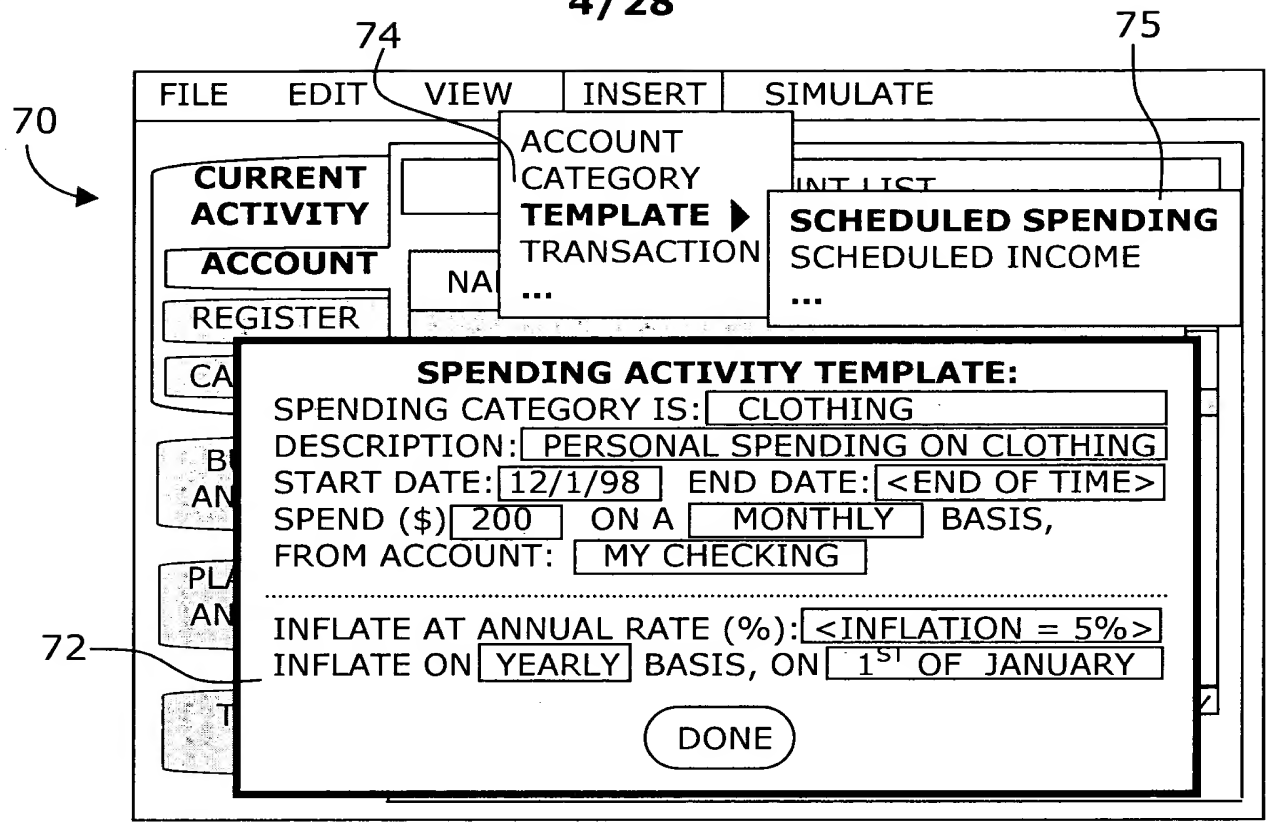


FIG. 7

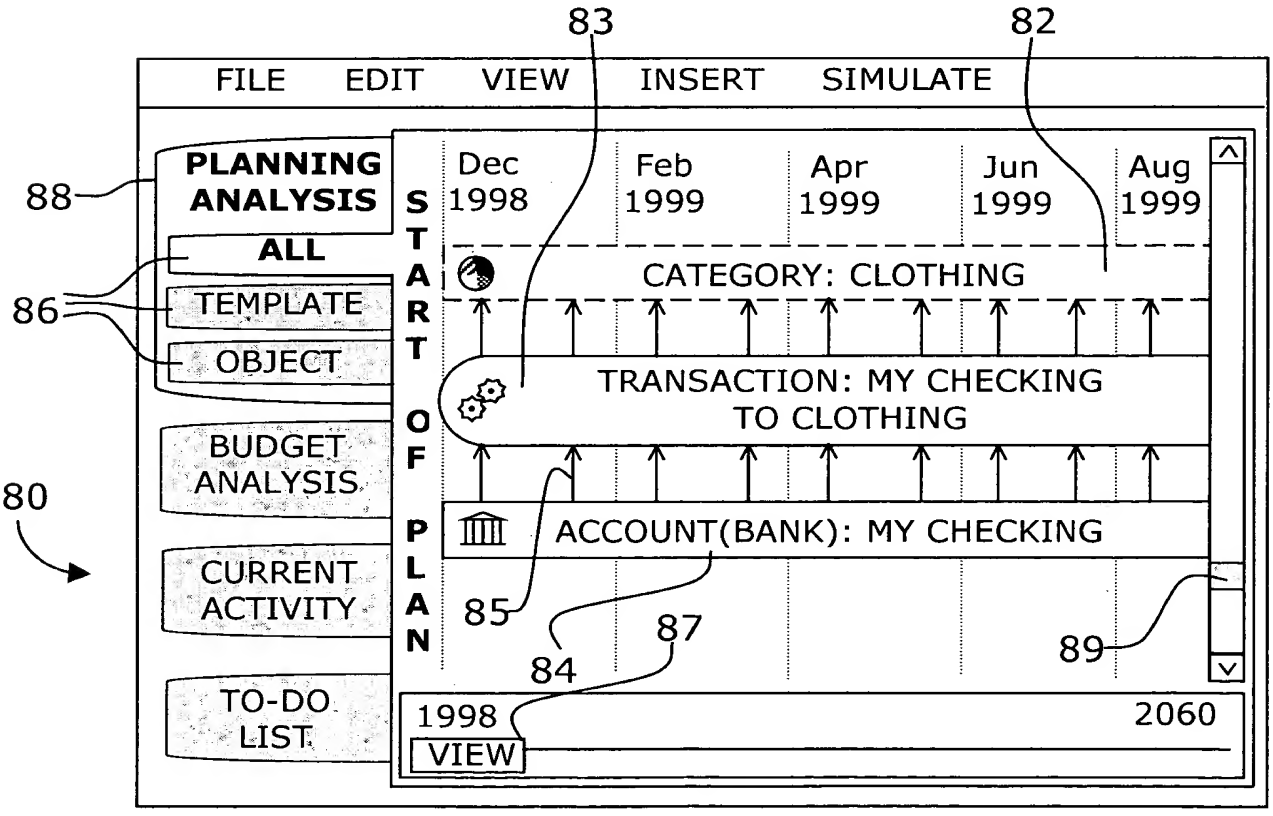


FIG. 8

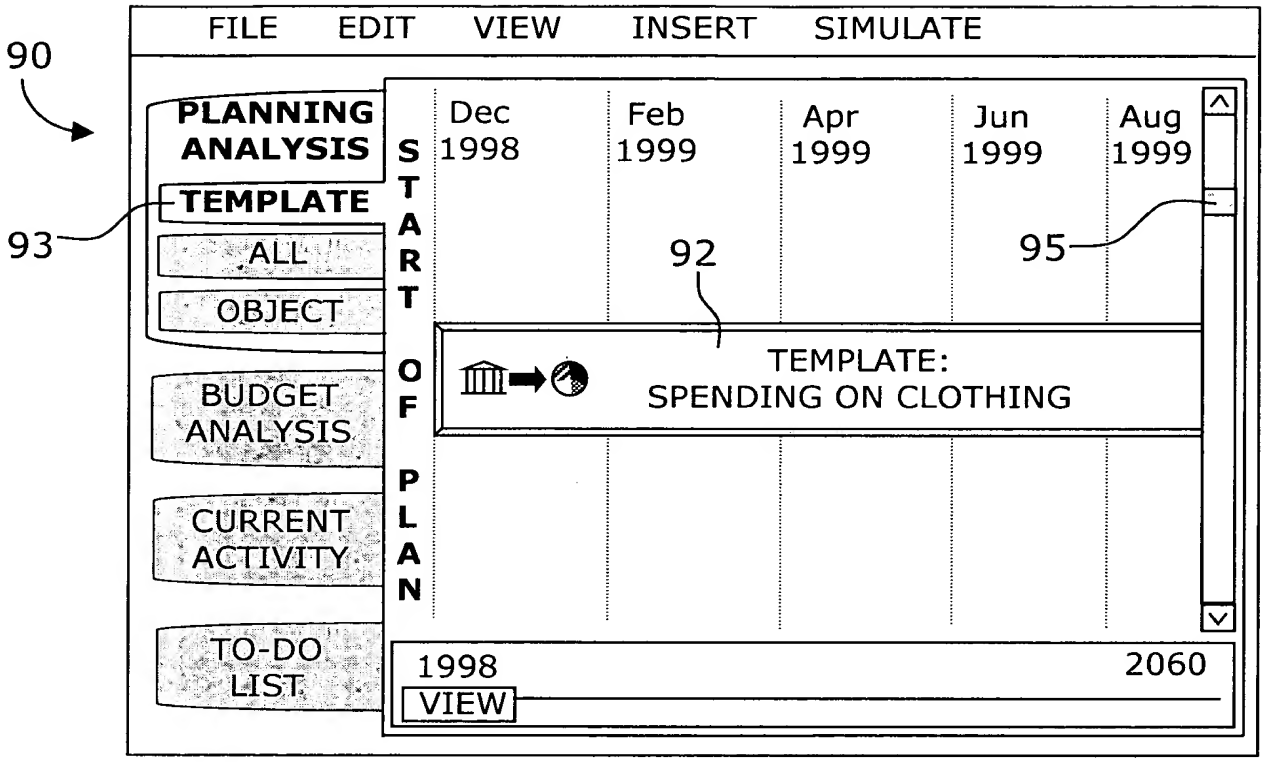


FIG. 9

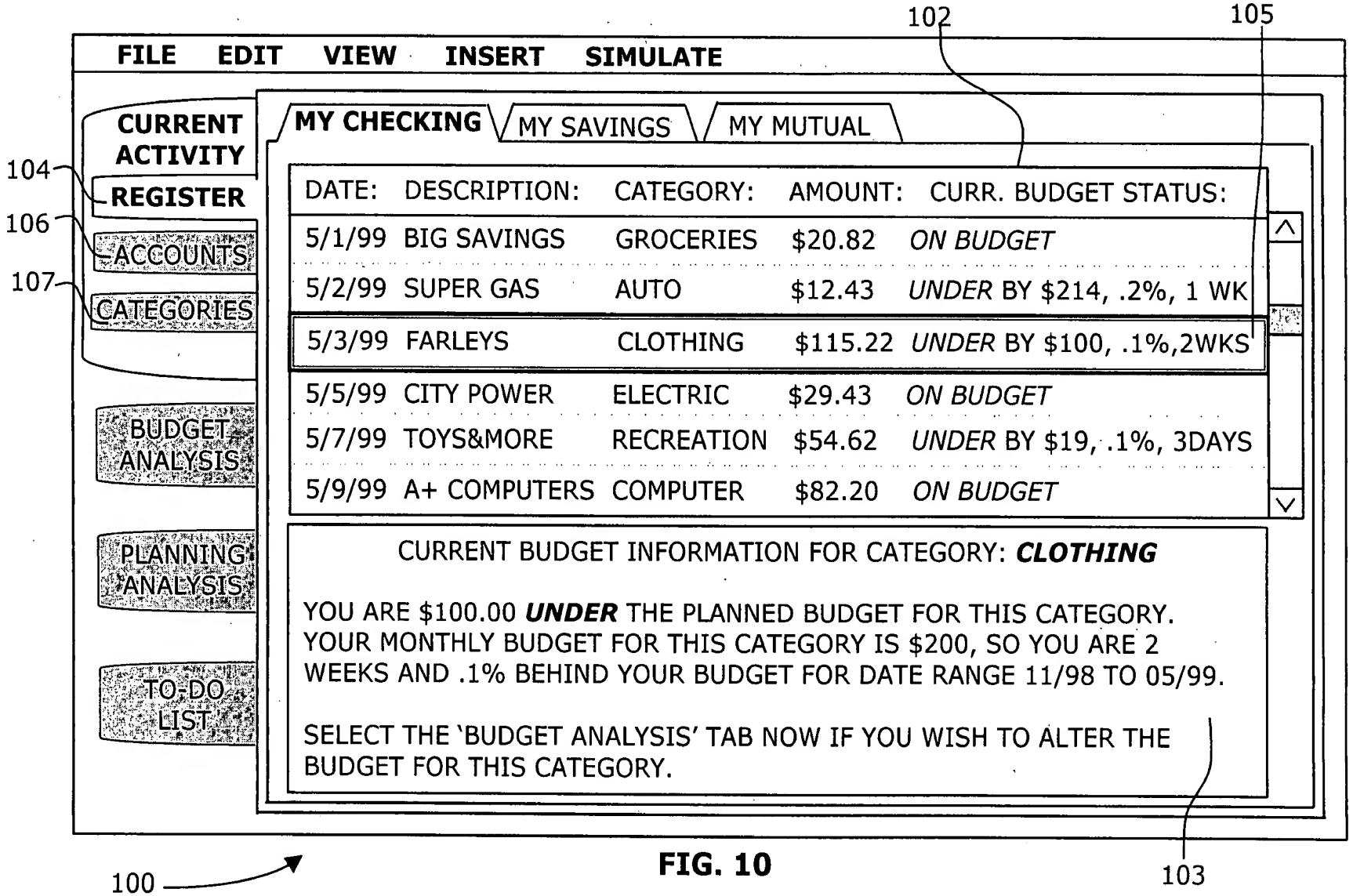


FIG. 10

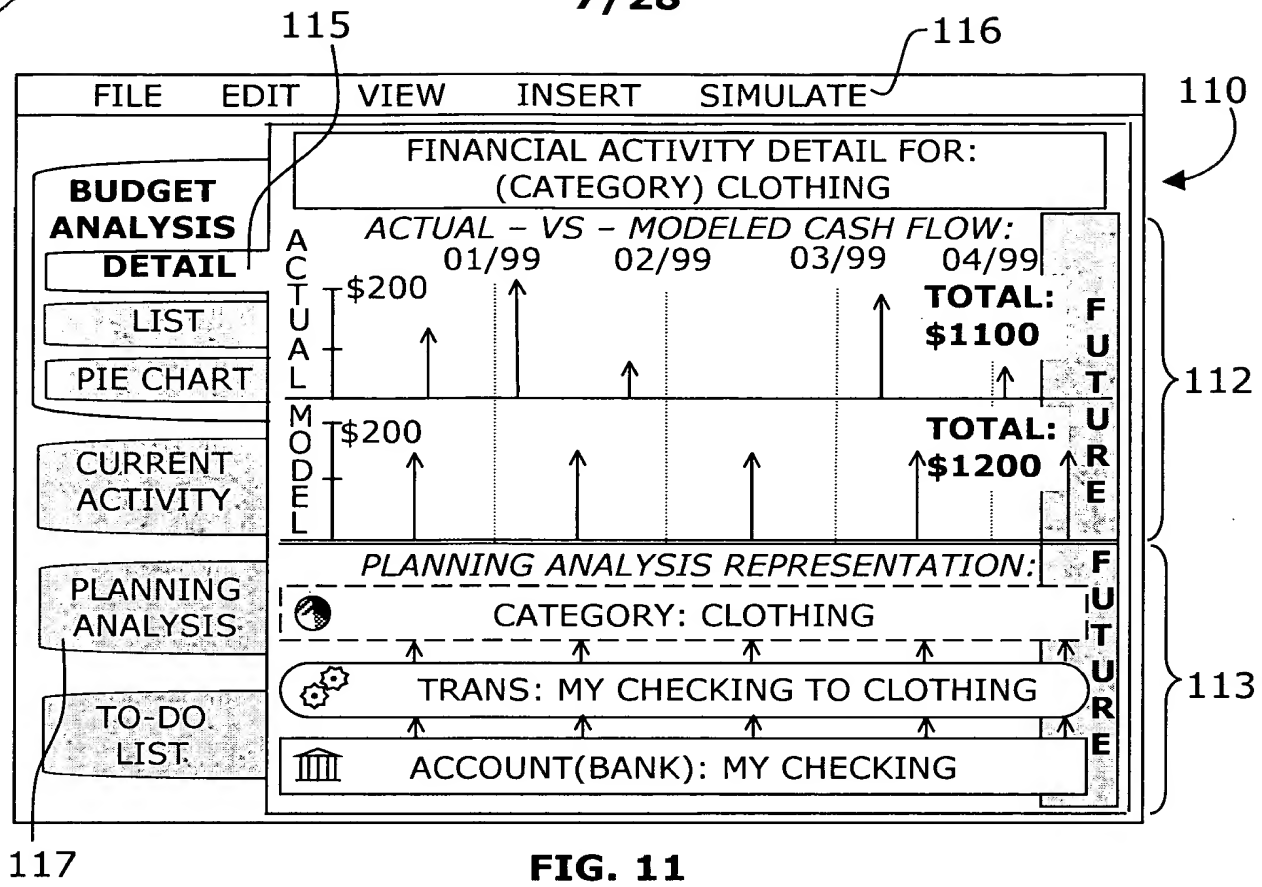


FIG. 11

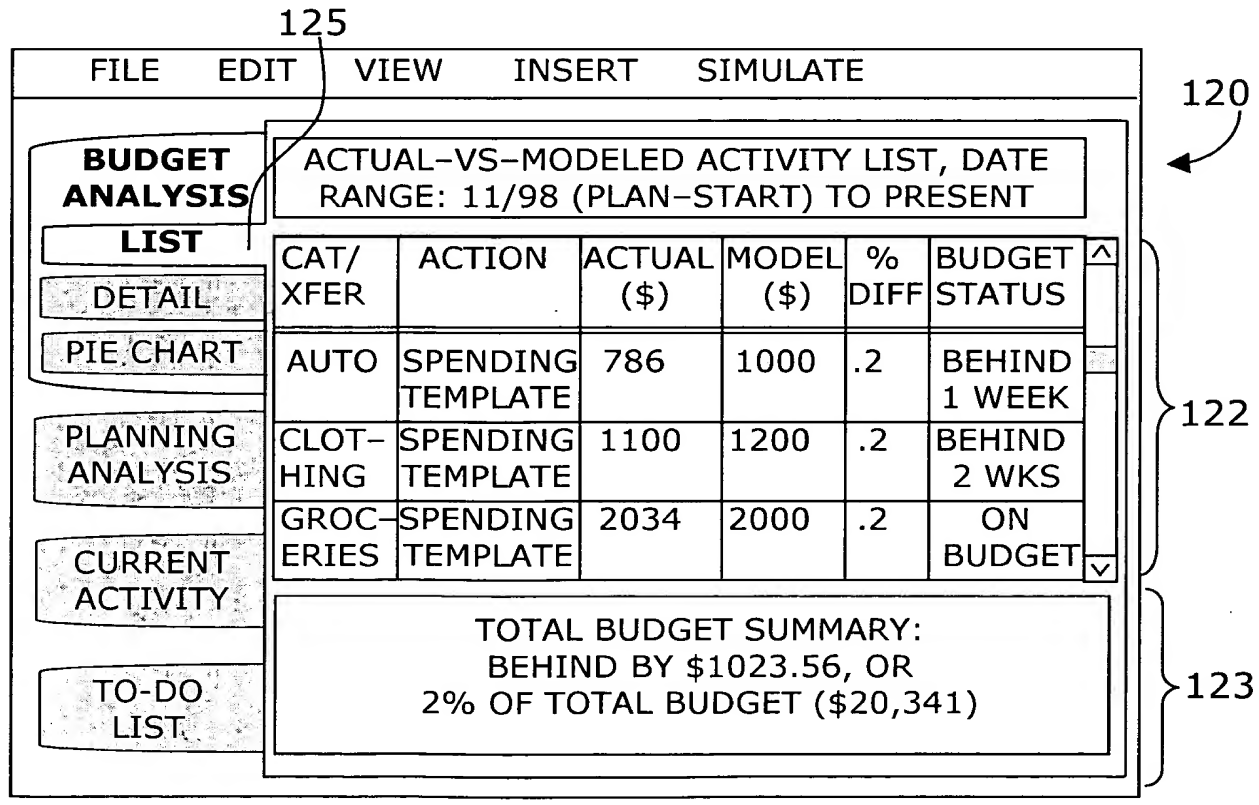


FIG. 12

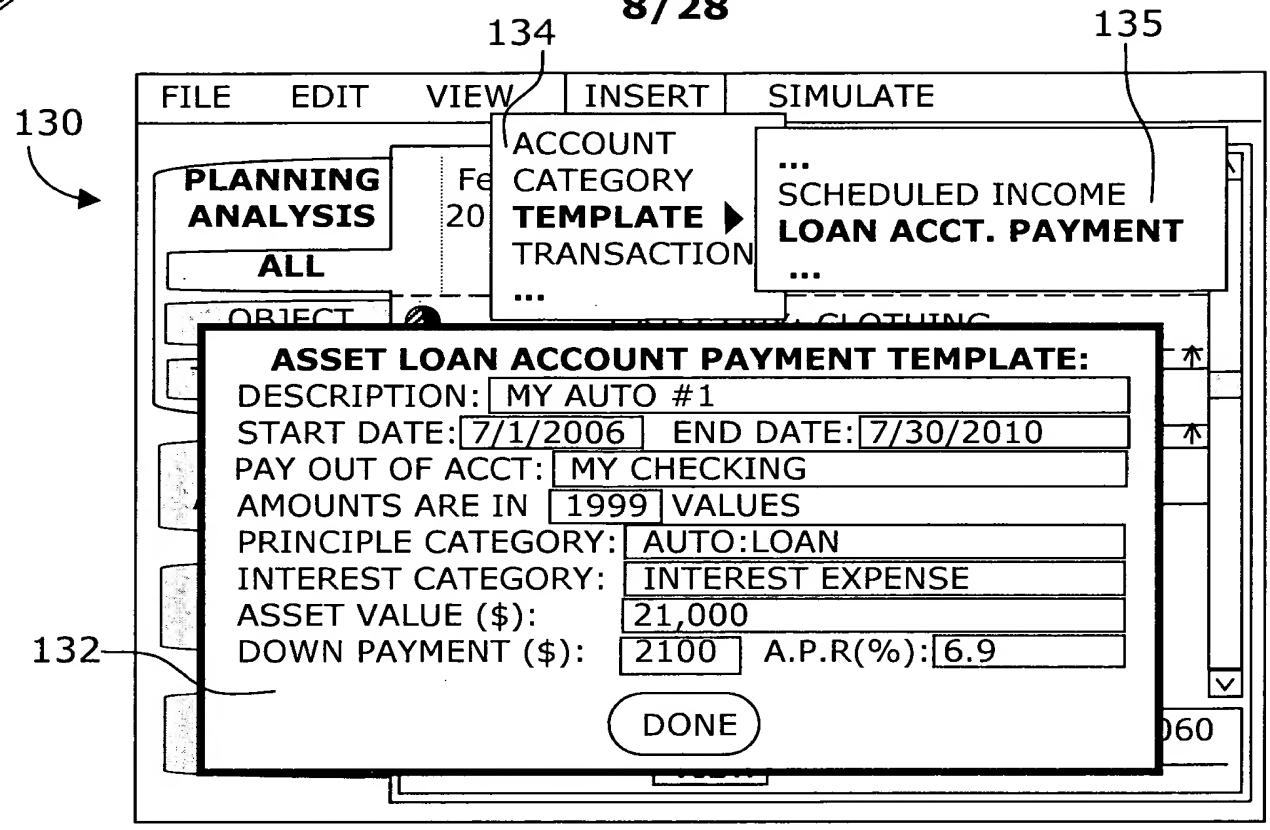


FIG. 13

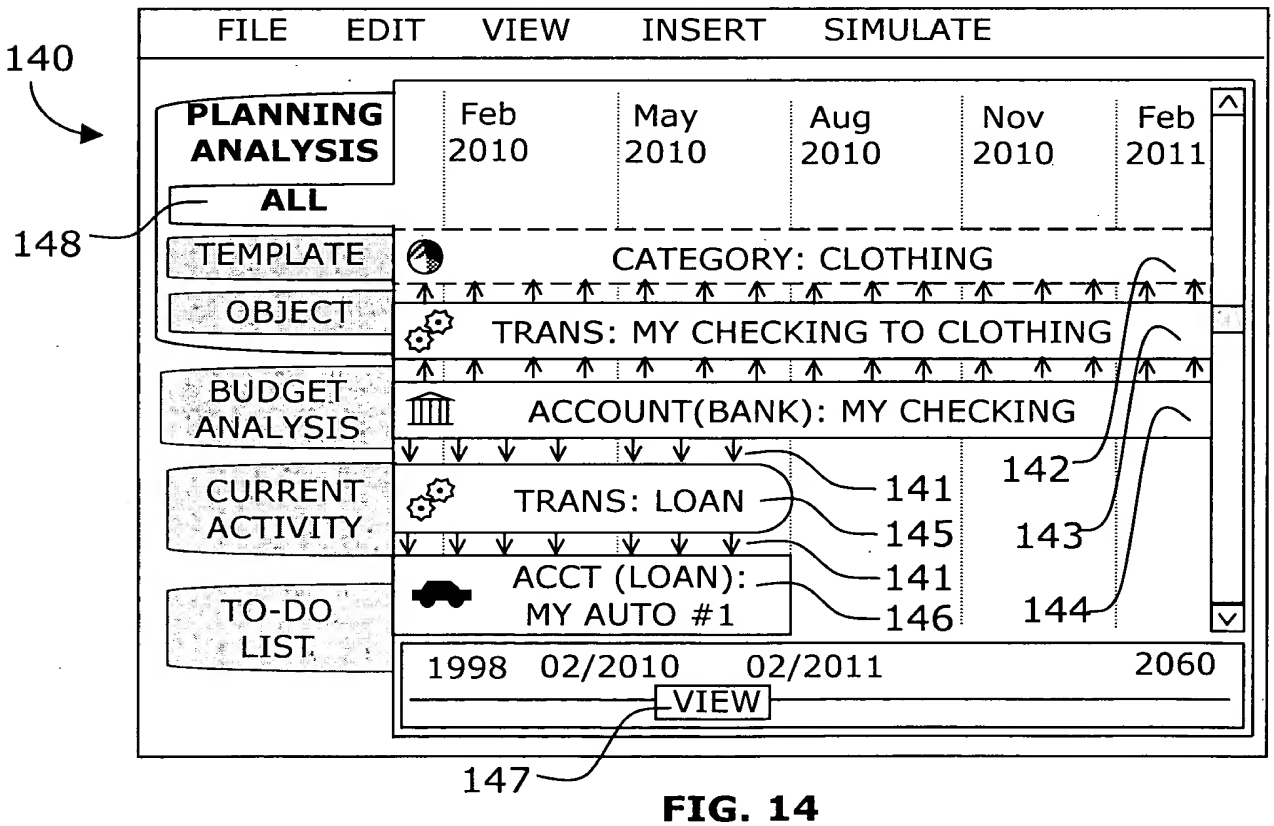


FIG. 14

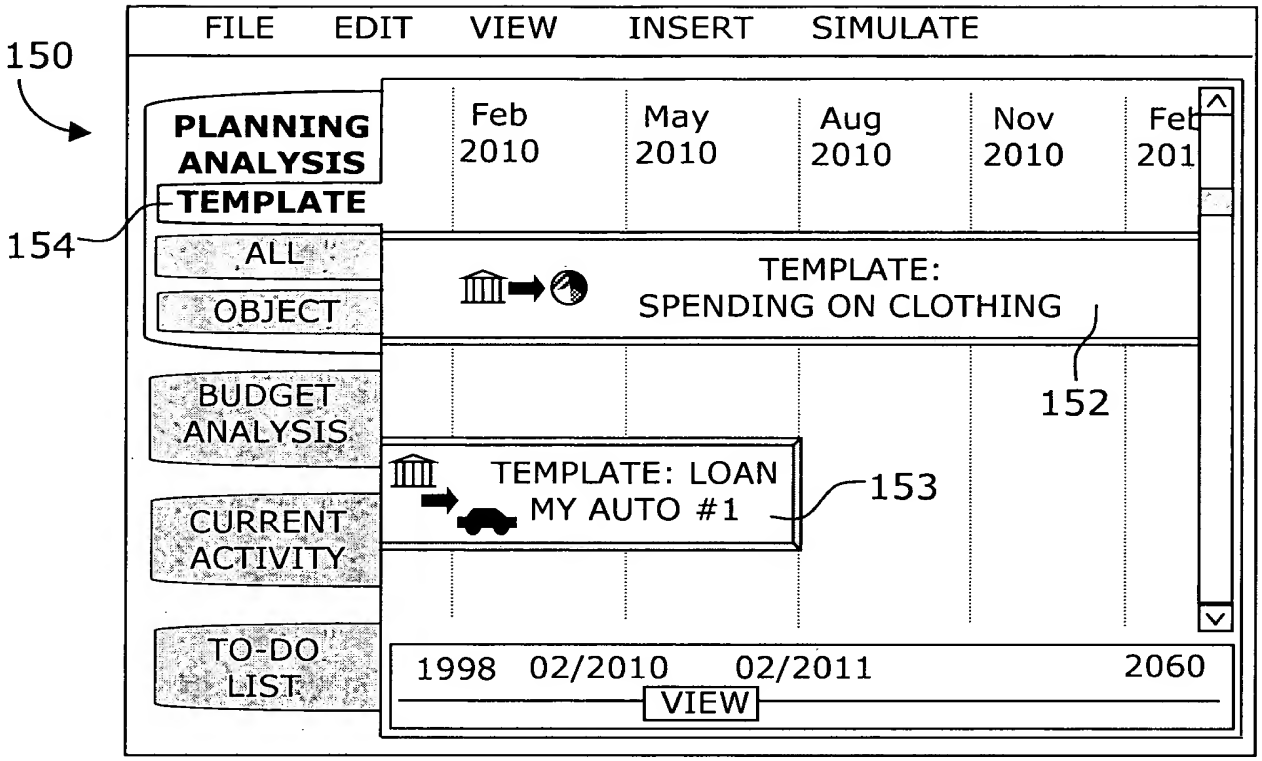


FIG. 15

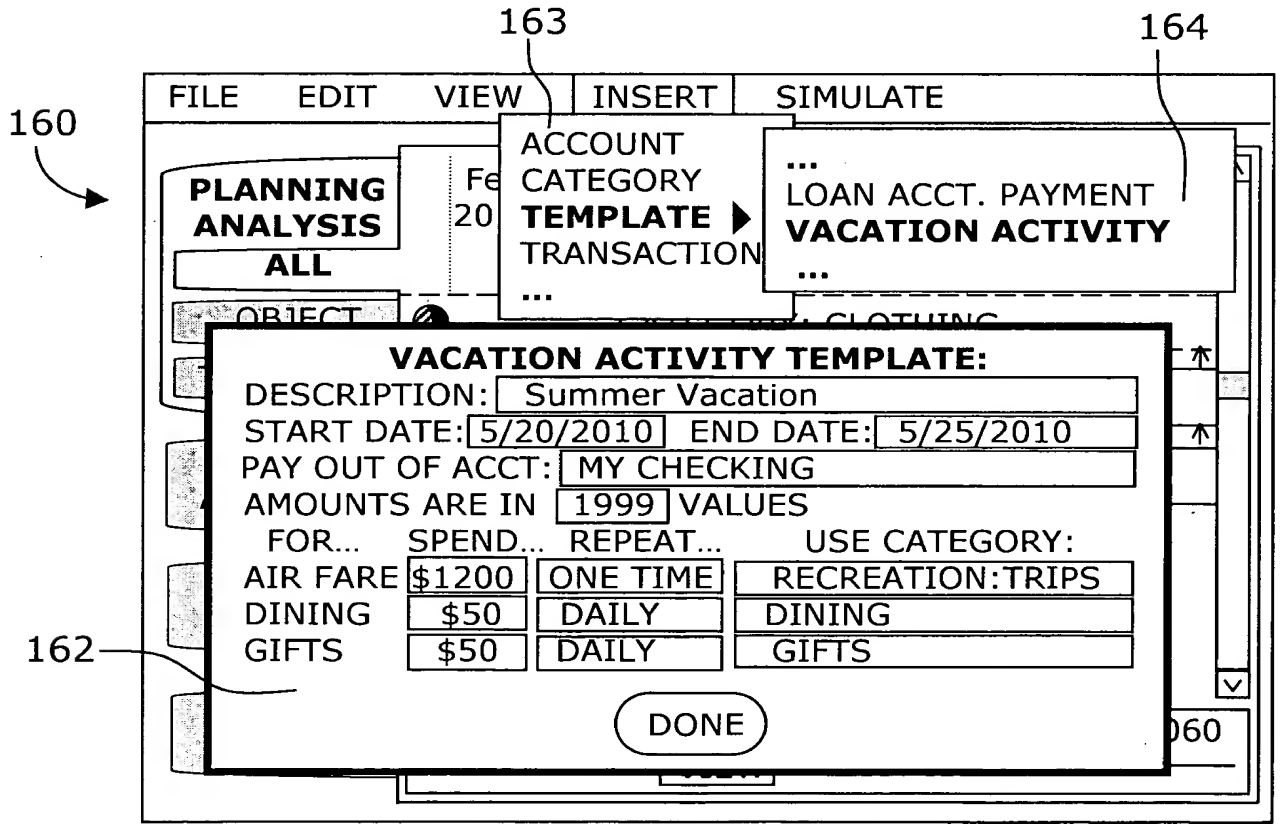


FIG. 16

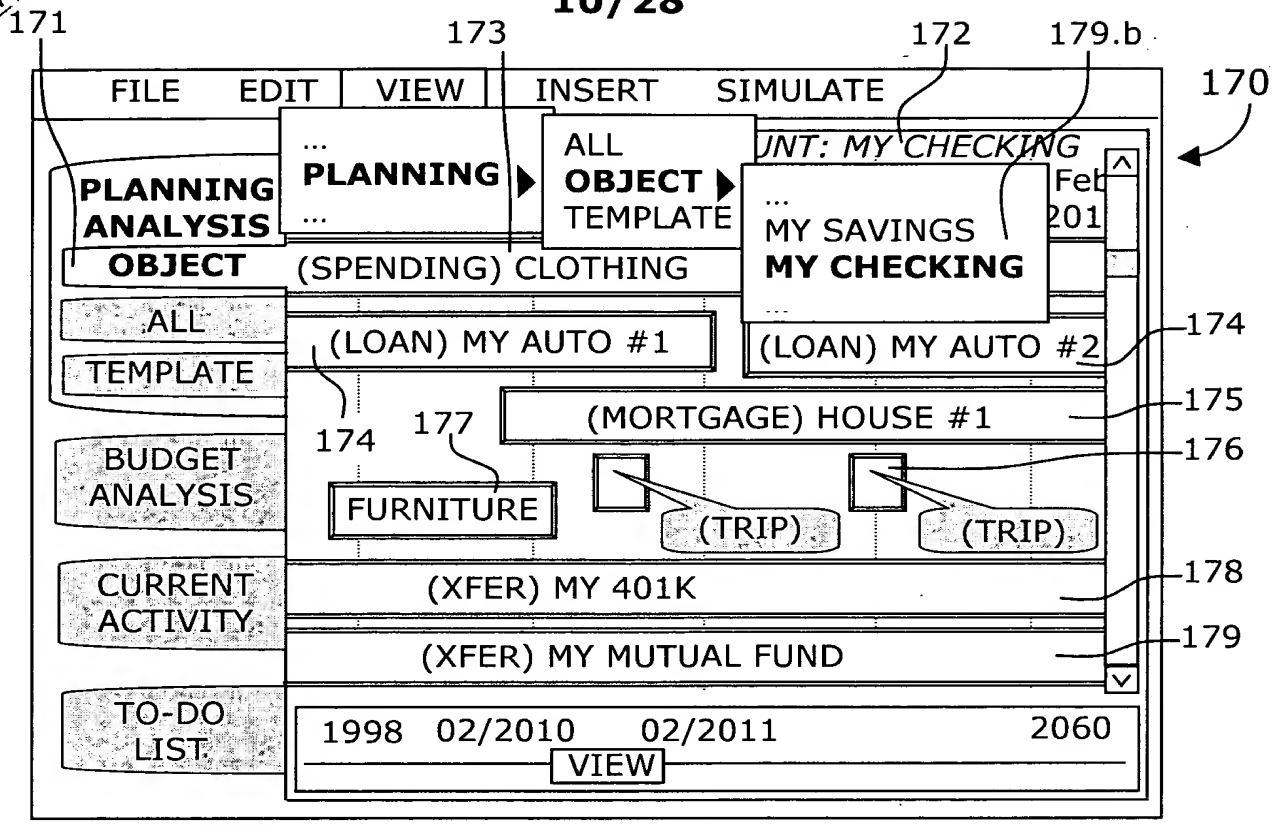


FIG. 17

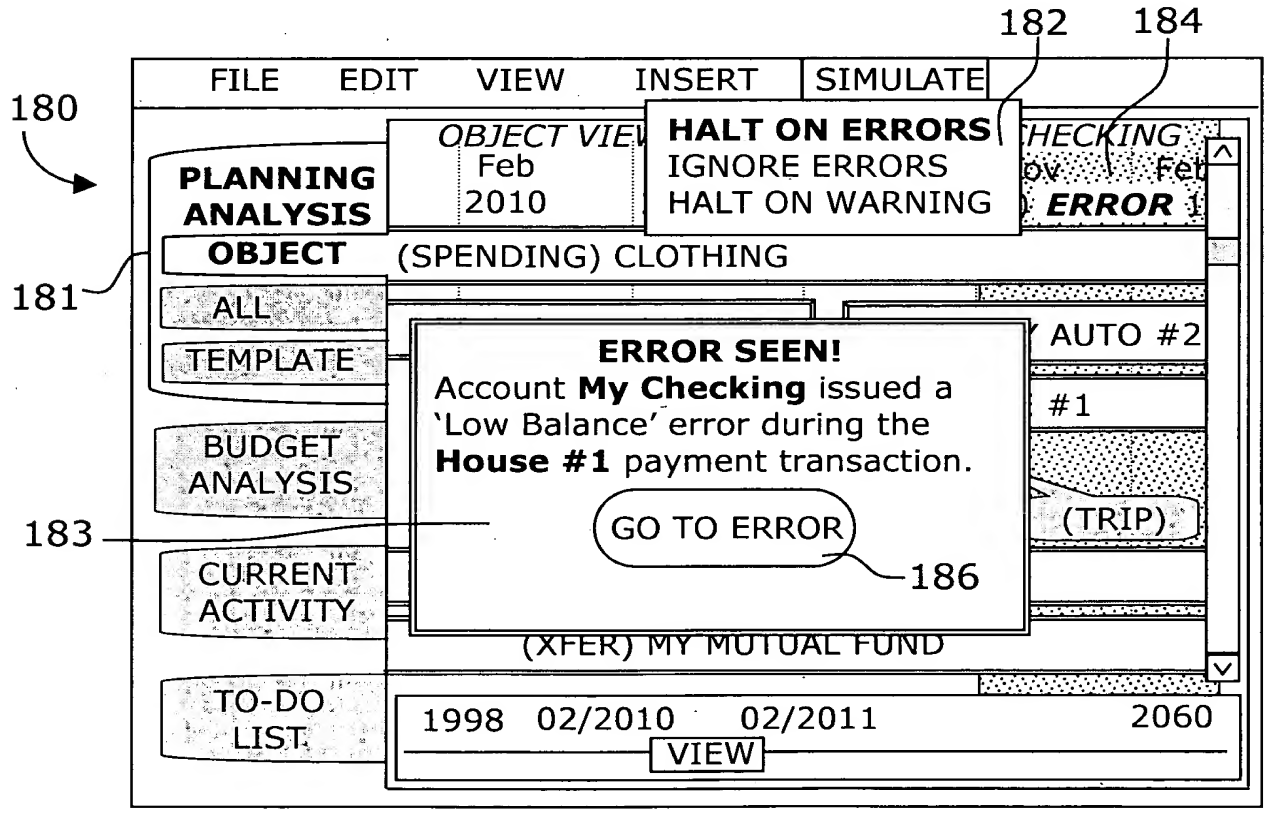


FIG. 18

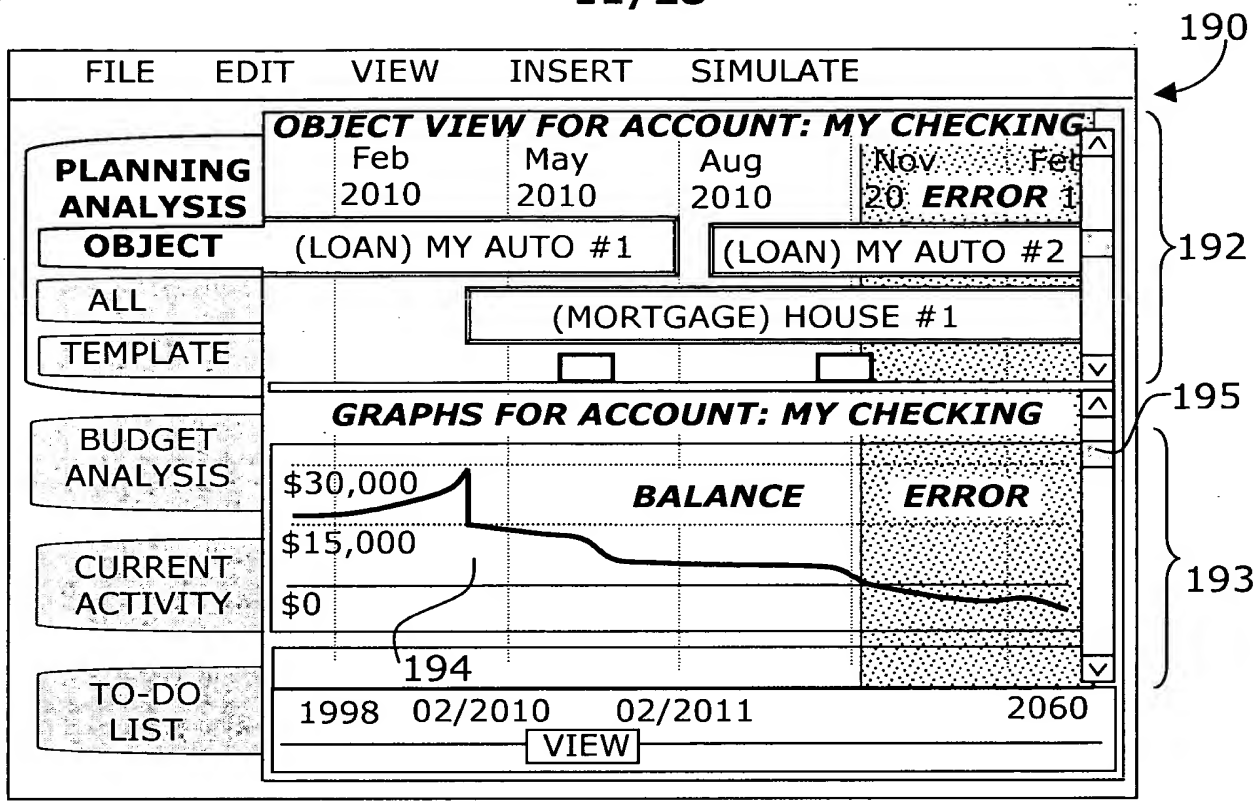


FIG. 19

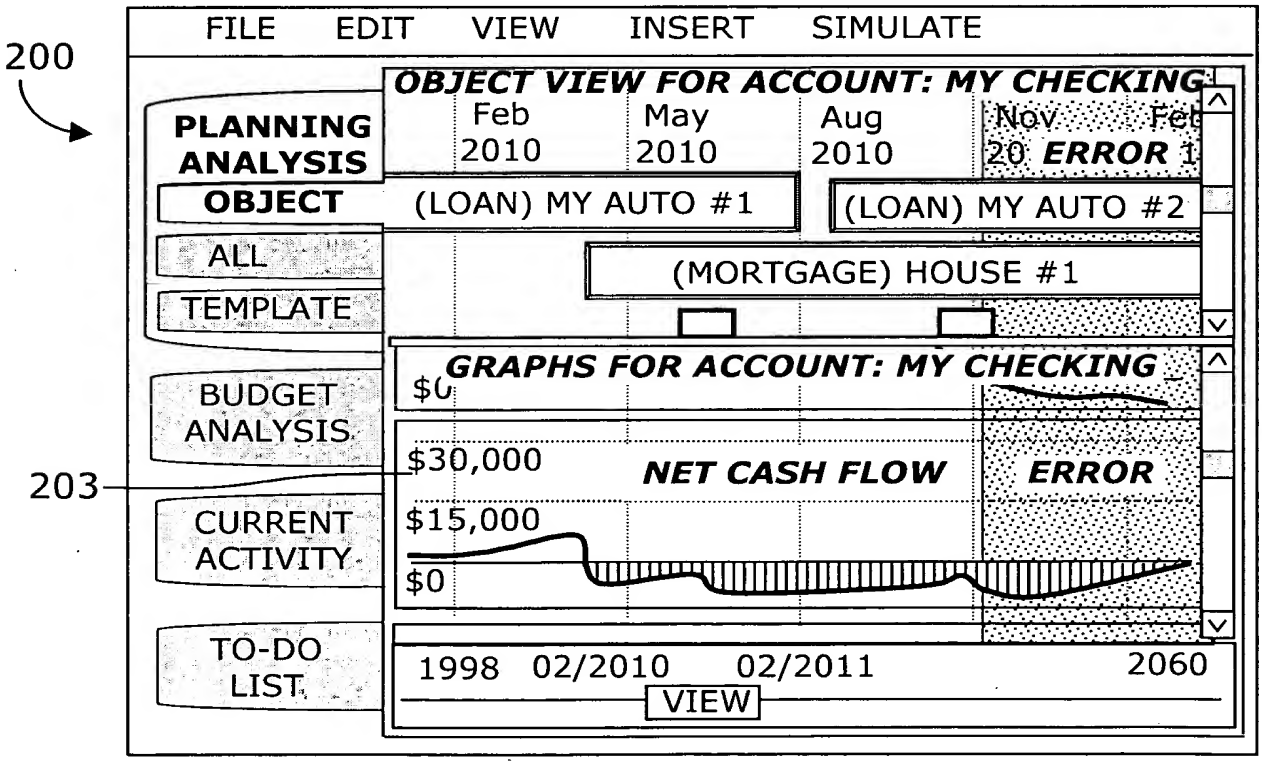


FIG. 20

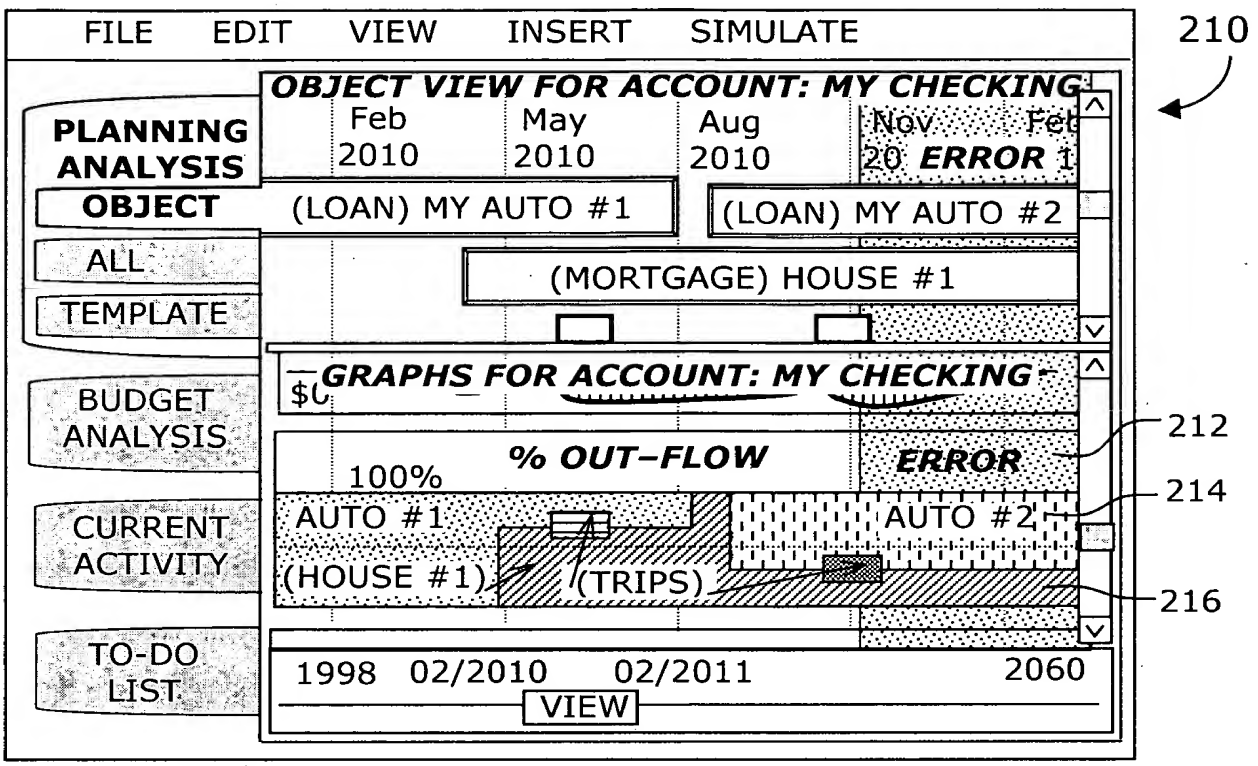


FIG. 21

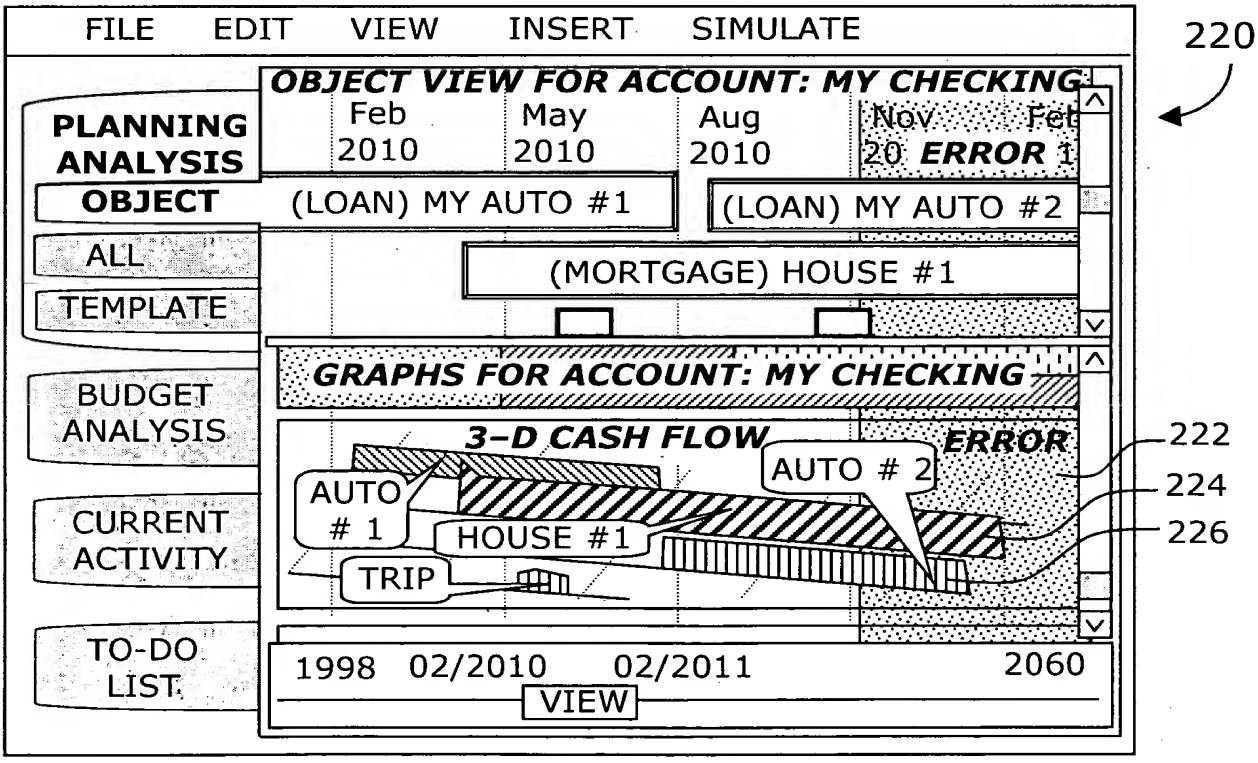


FIG. 22

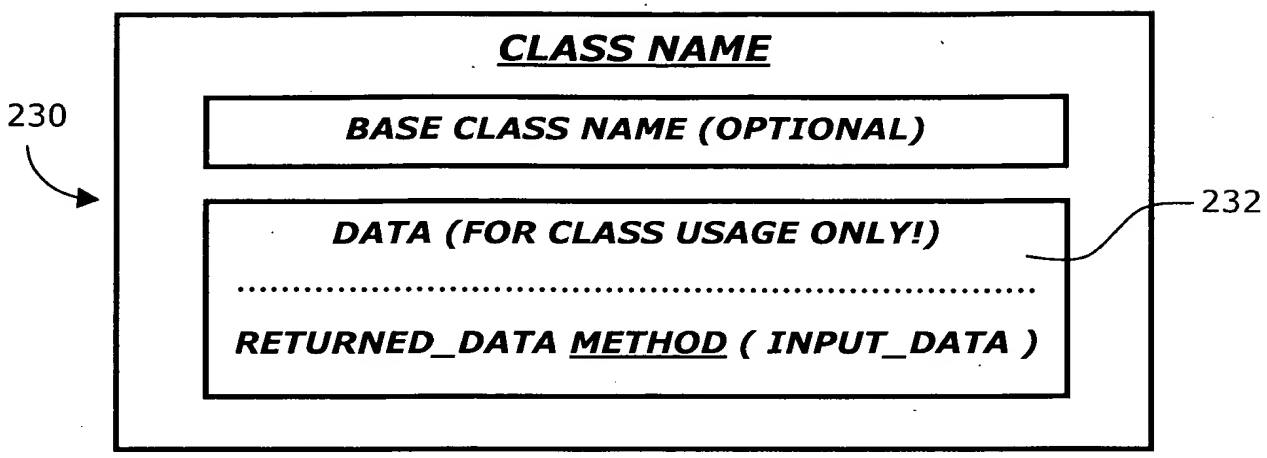


FIG. 23

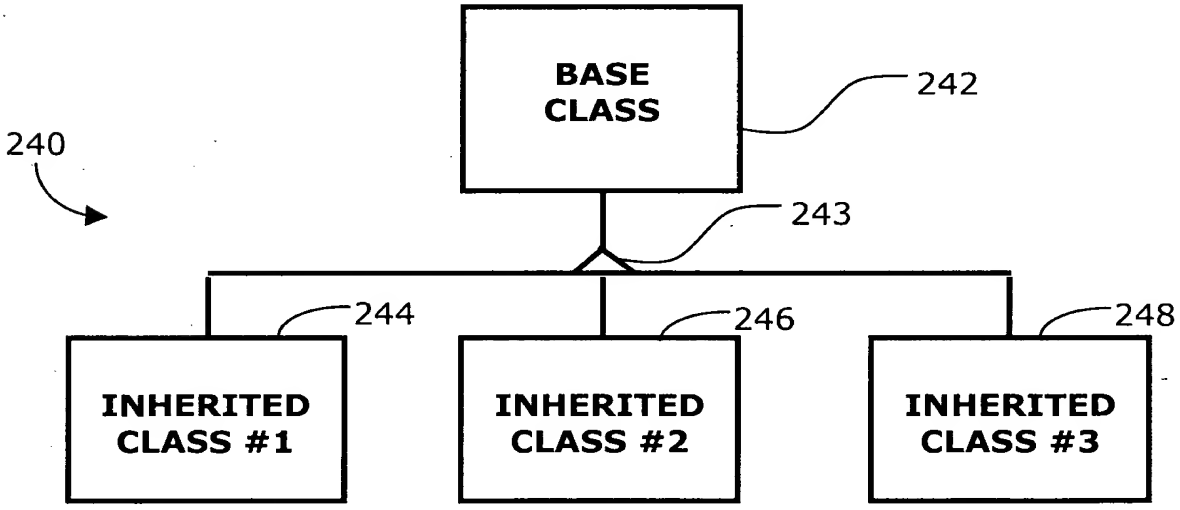


FIG. 24

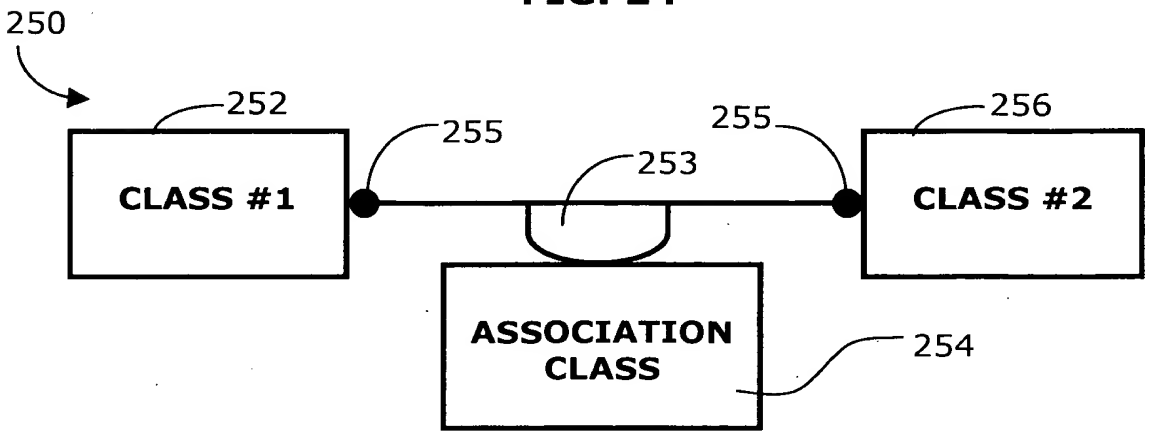


FIG. 25

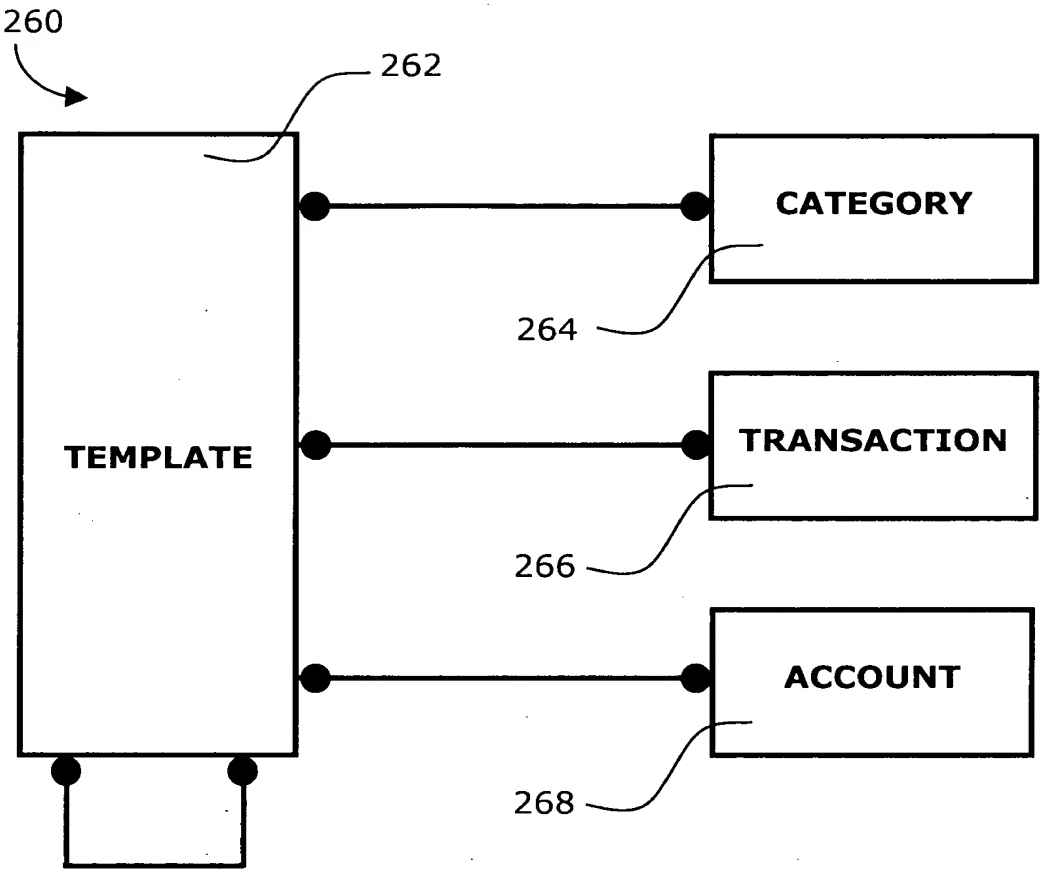


FIG. 26

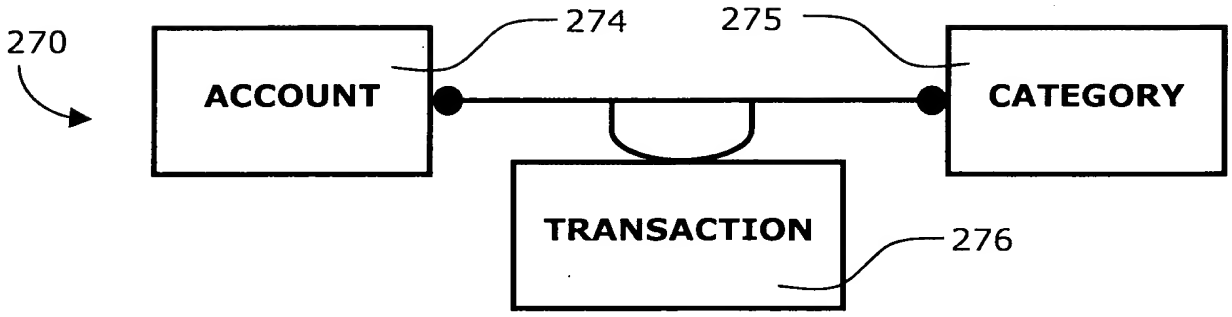


FIG. 27A

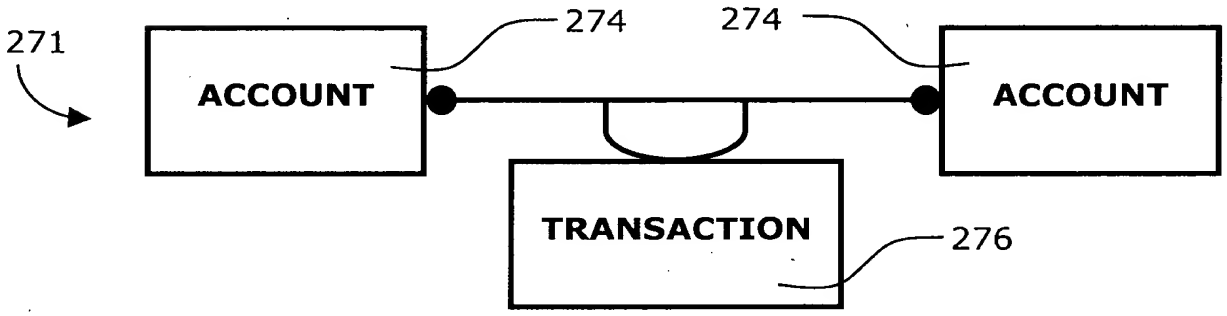


FIG. 27B

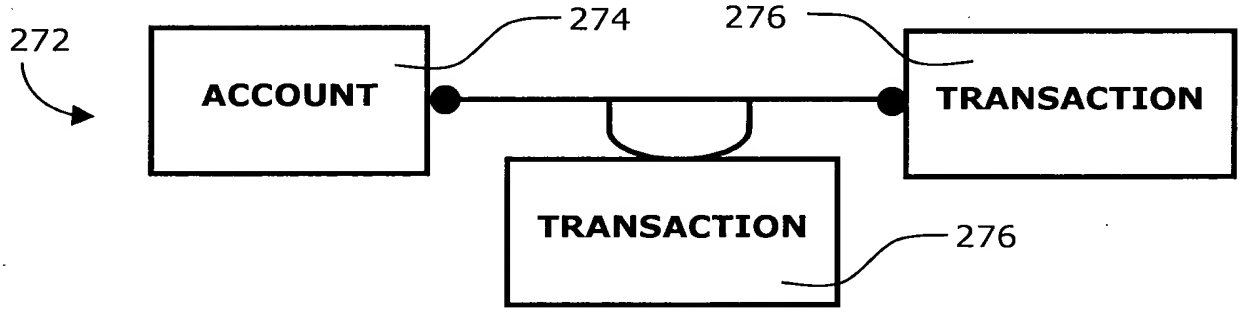


FIG. 27C

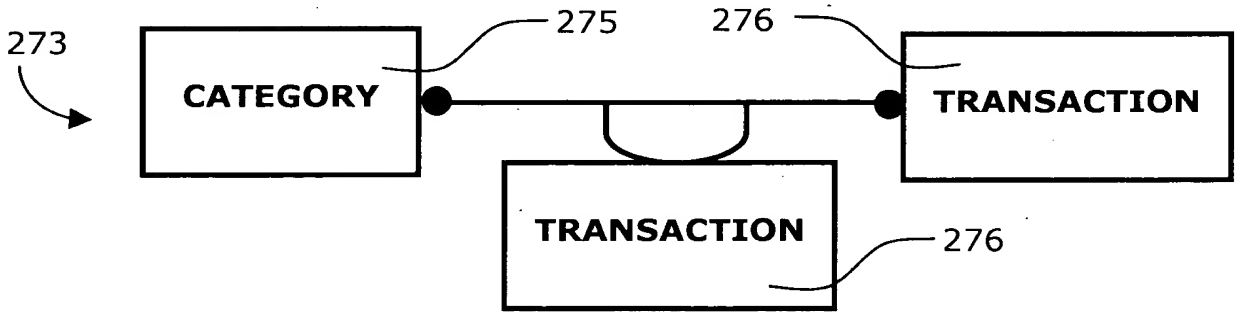


FIG. 27D

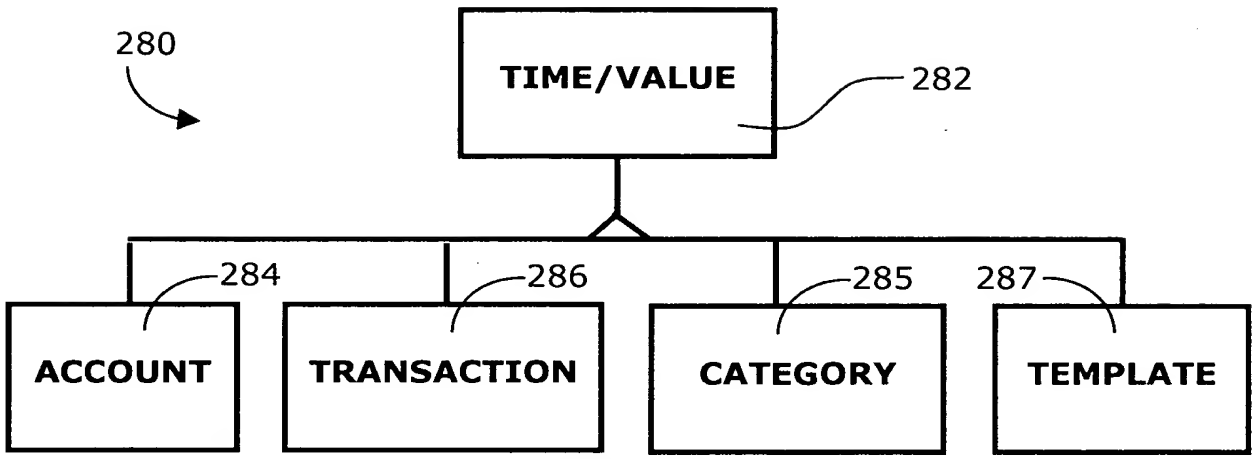


FIG. 28

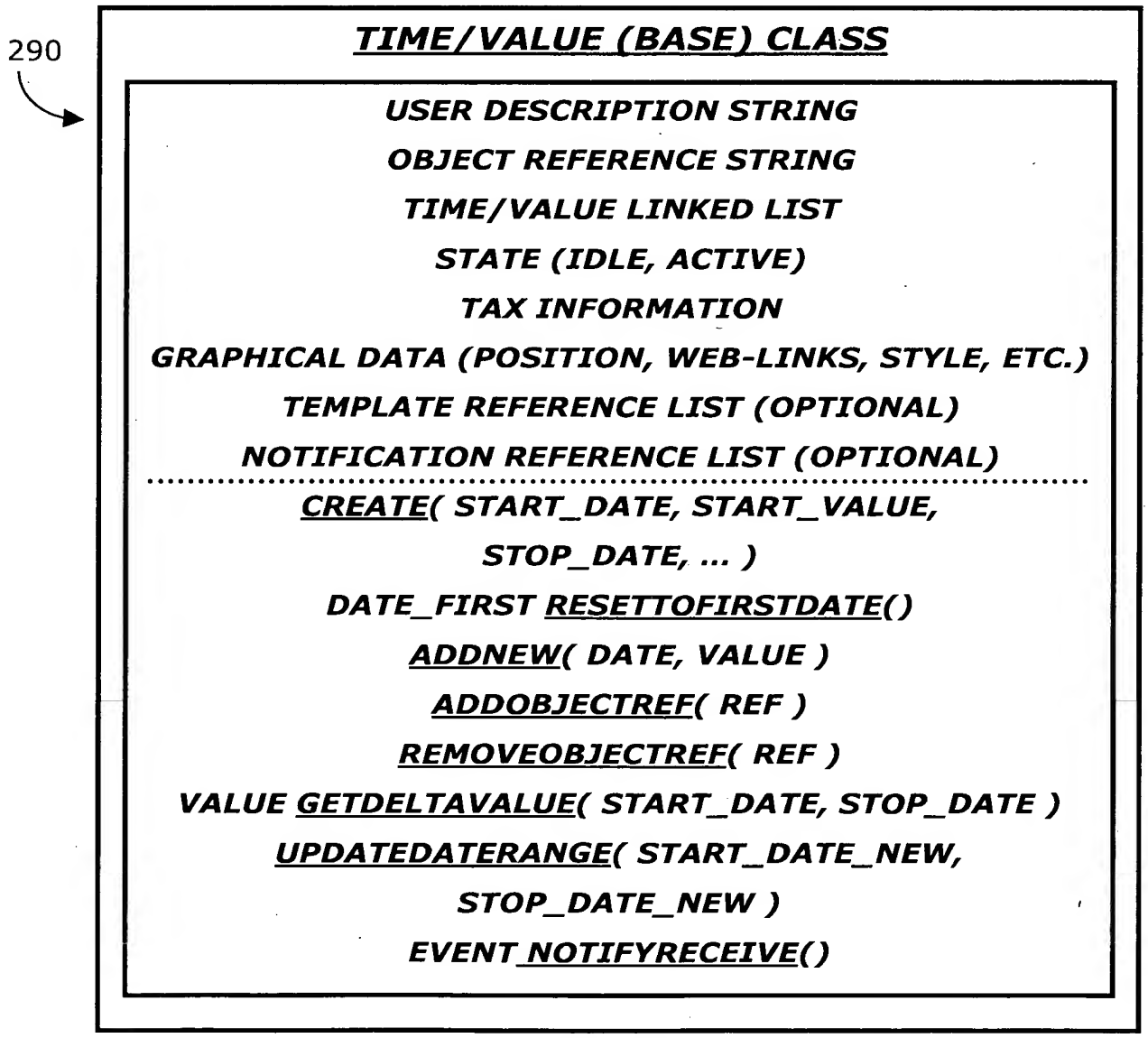


FIG. 29

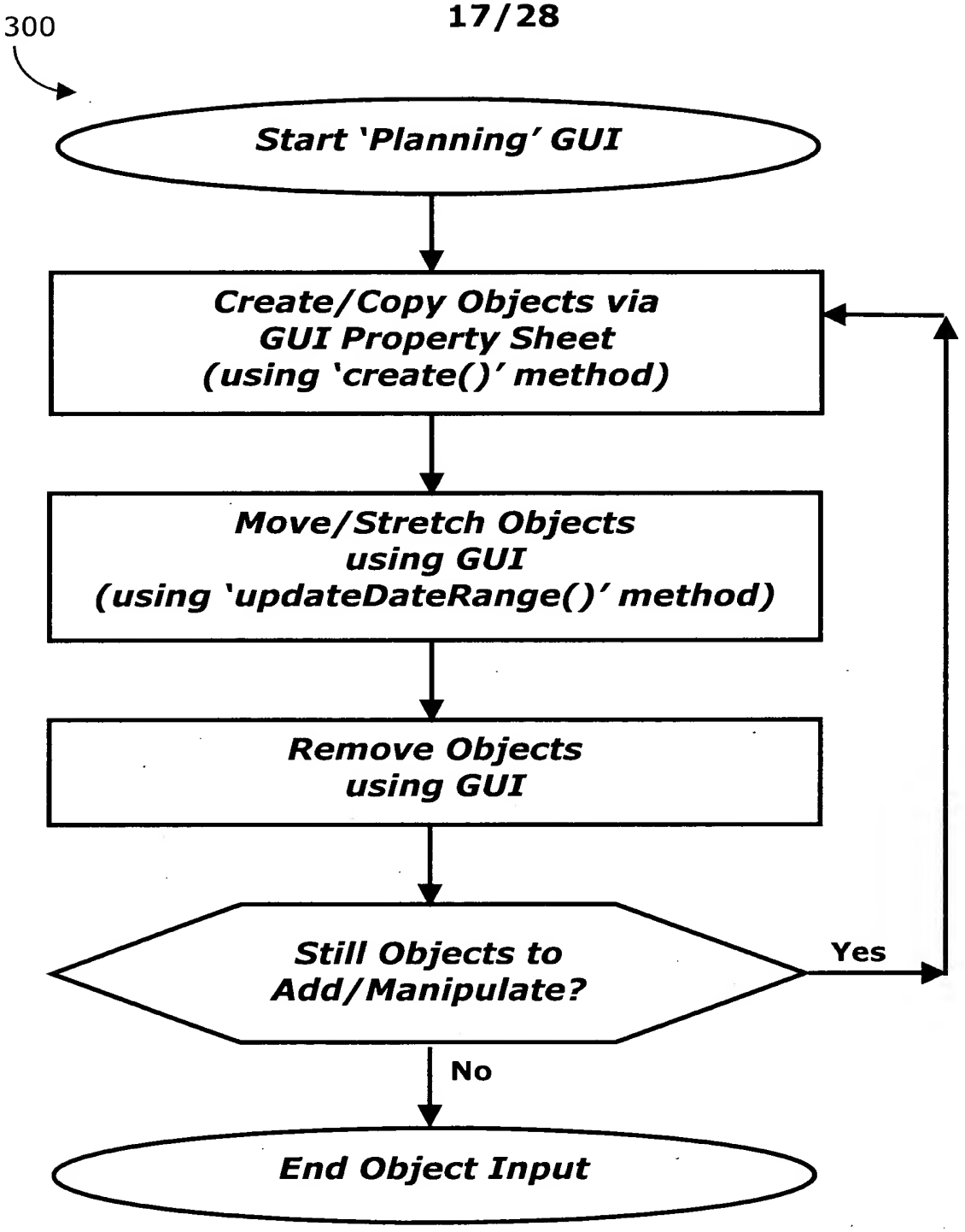


FIG. 30

310

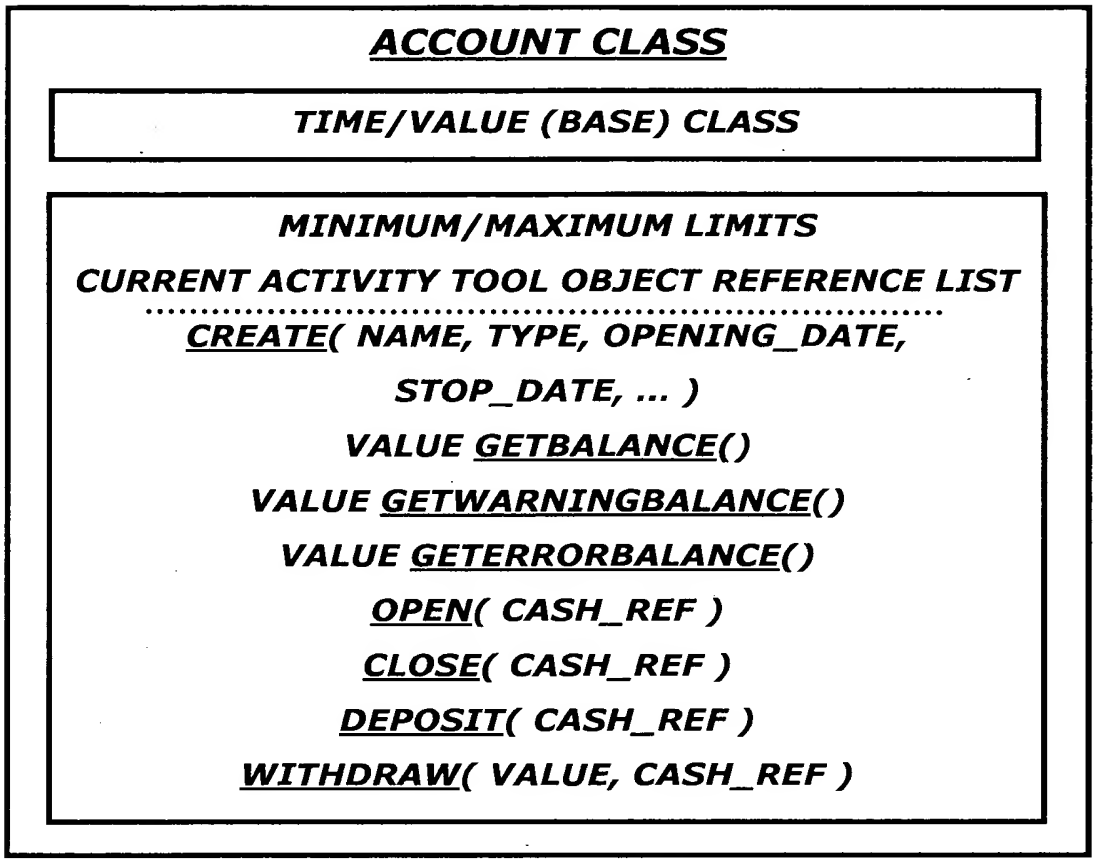


FIG. 31

320

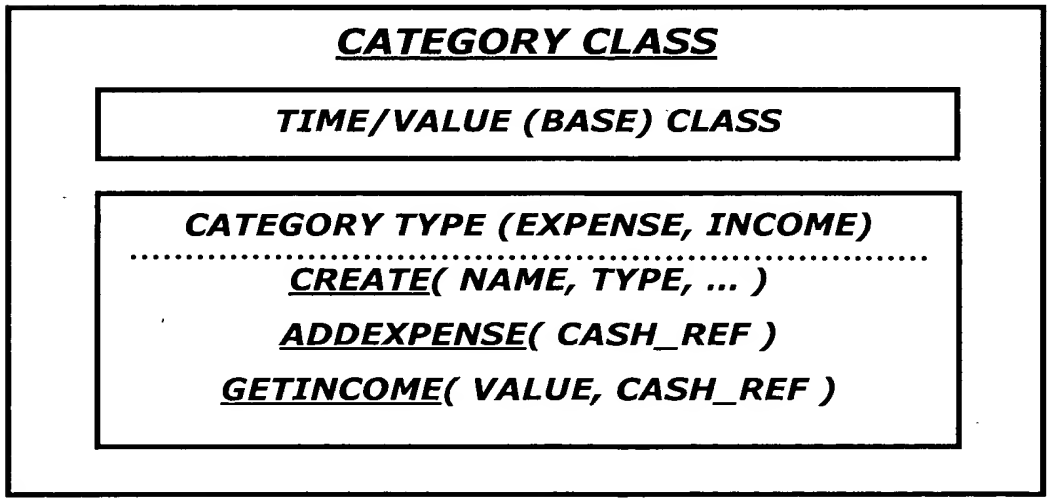


FIG. 32

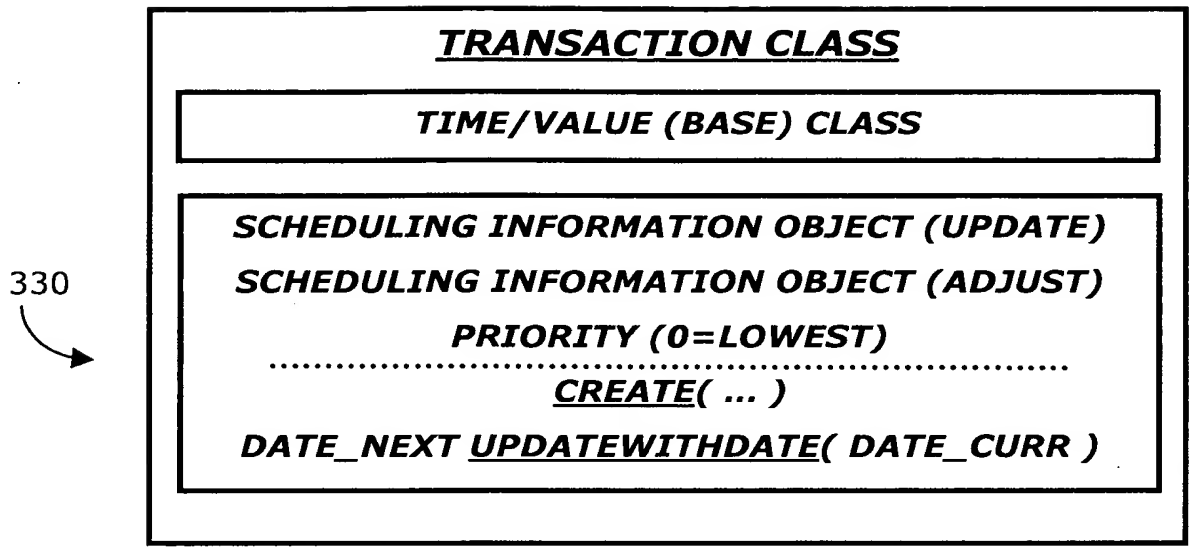


FIG. 33

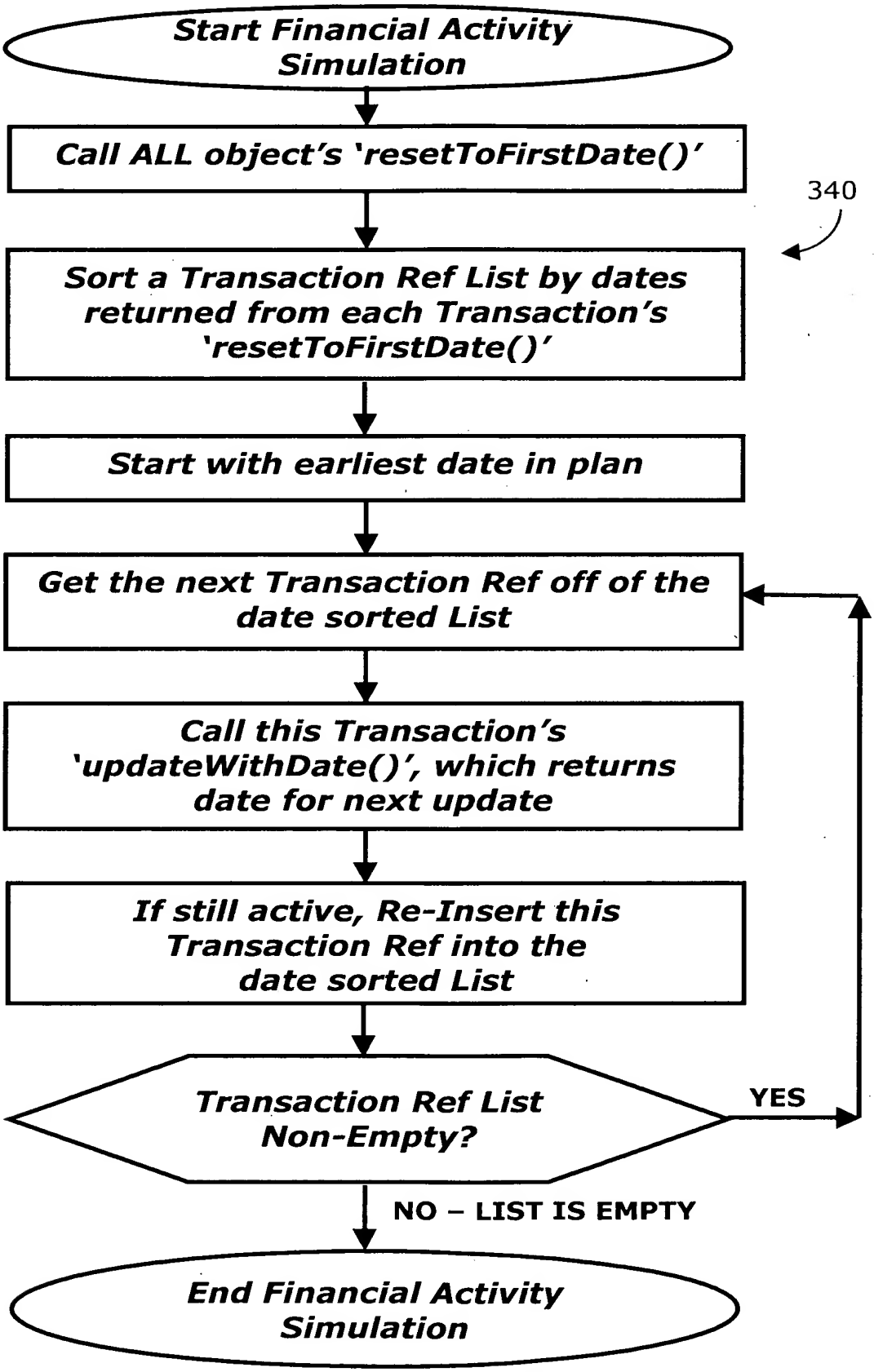


FIG. 34

350

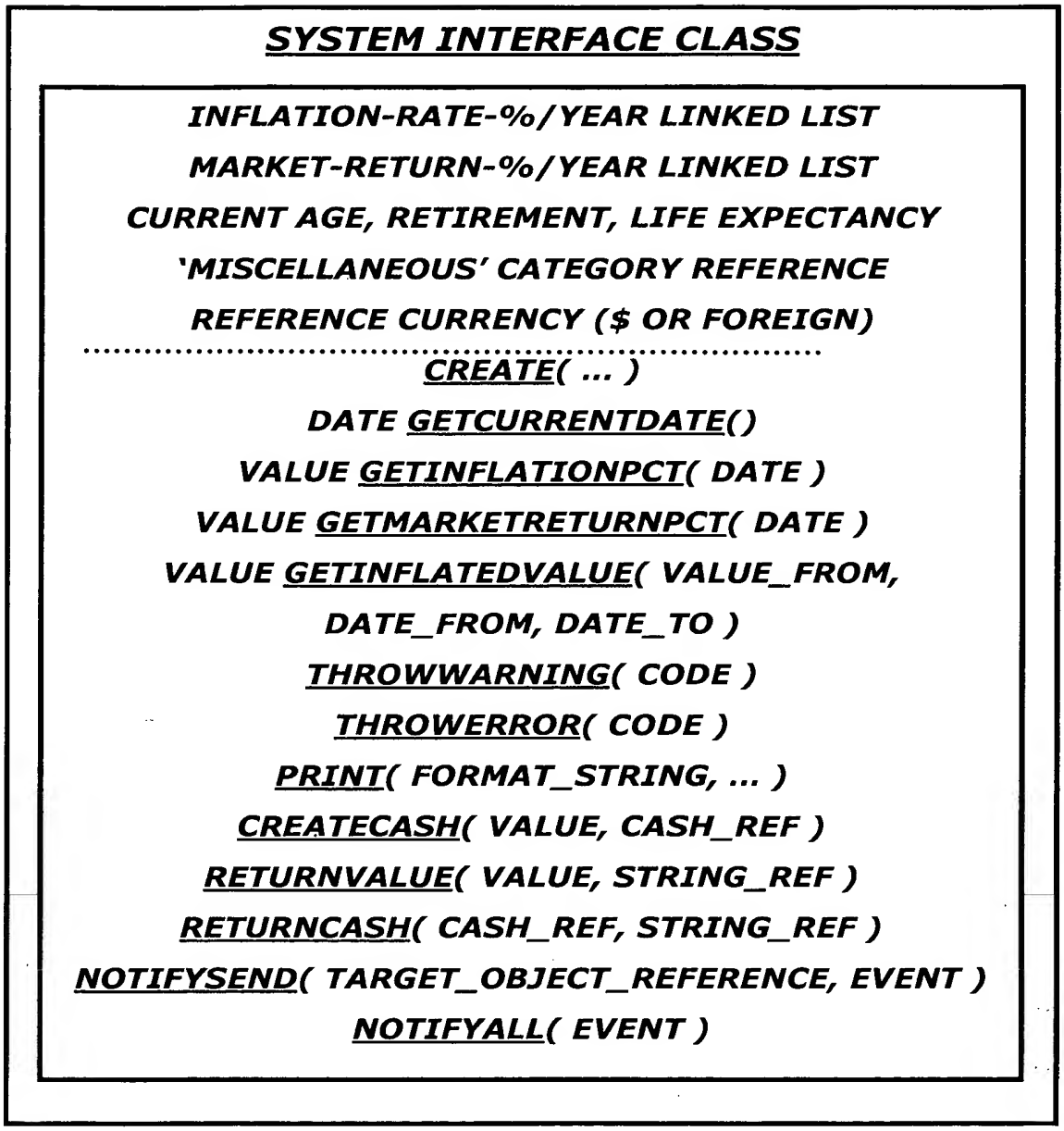


FIG. 35

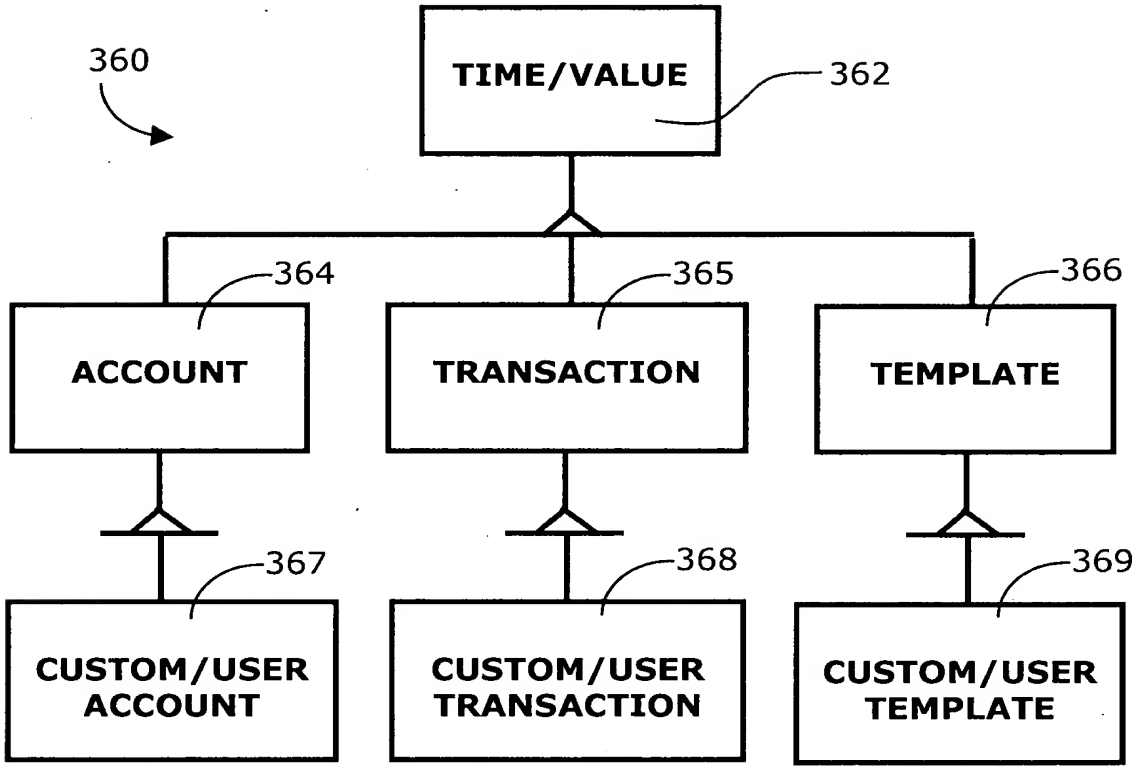


FIG. 36

370

SCHEDULING INFORMATION CLASS

FIRST DATE

LAST DATE

NEXT SCHEDULED DATE

SCHEDULING METHOD (DAILY, WEEKLY, MONTHLY, ETC.)

SCHEDULING FREQUENCY (EVERY TIME, EVERY OTHER TIME, EVERY 3RD, ETC.)

CREATE(...)

DATE RESETTOFIRSTDATE()

DATE GETNEXTDATE()

SETNEXTDATE(DATE)

DATE COMPUTENEXTDATE()

UPDATEDATERANGE(START_DATE_NEW, STOP_DATE_NEW)

FIG. 37

380

ACCOUNT-TO-ACCOUNT TRANSFER TRANSACTION CLASS

TRANSACTION (BASE) CLASS

'FROM' ACCOUNT OBJECT REFERENCE

'TO' ACCOUNT #2 OBJECT REFERENCE

TRANSFER AMOUNT VALUE

ADJUSTMENT PERCENTAGE

CREATE(...)

FIG. 38

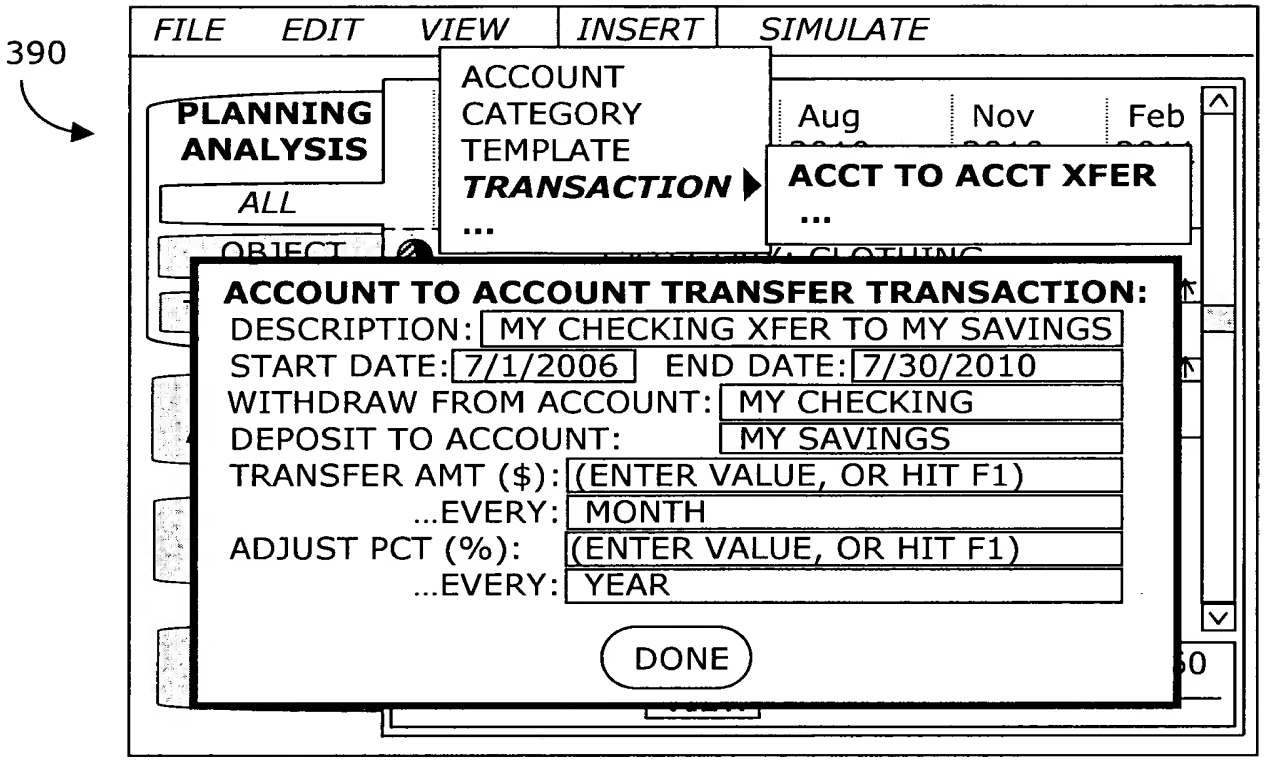


FIG. 39

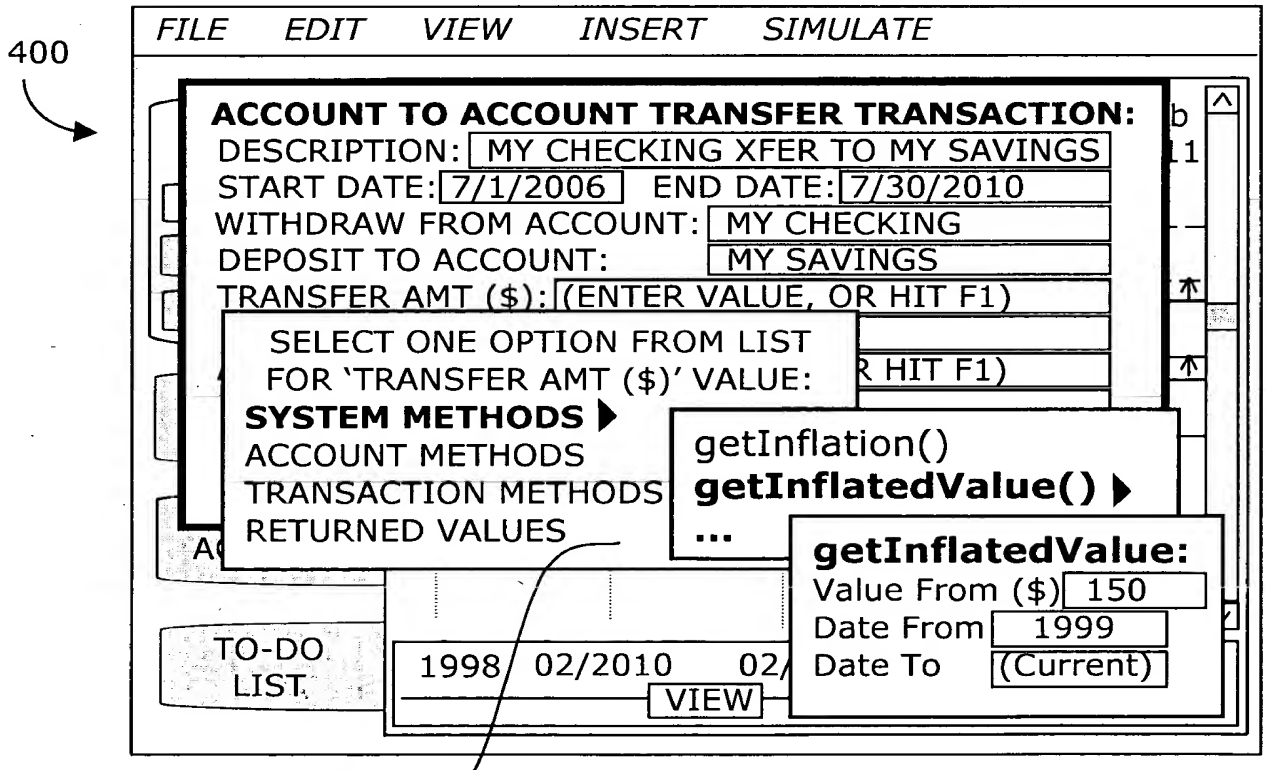


FIG. 40

410

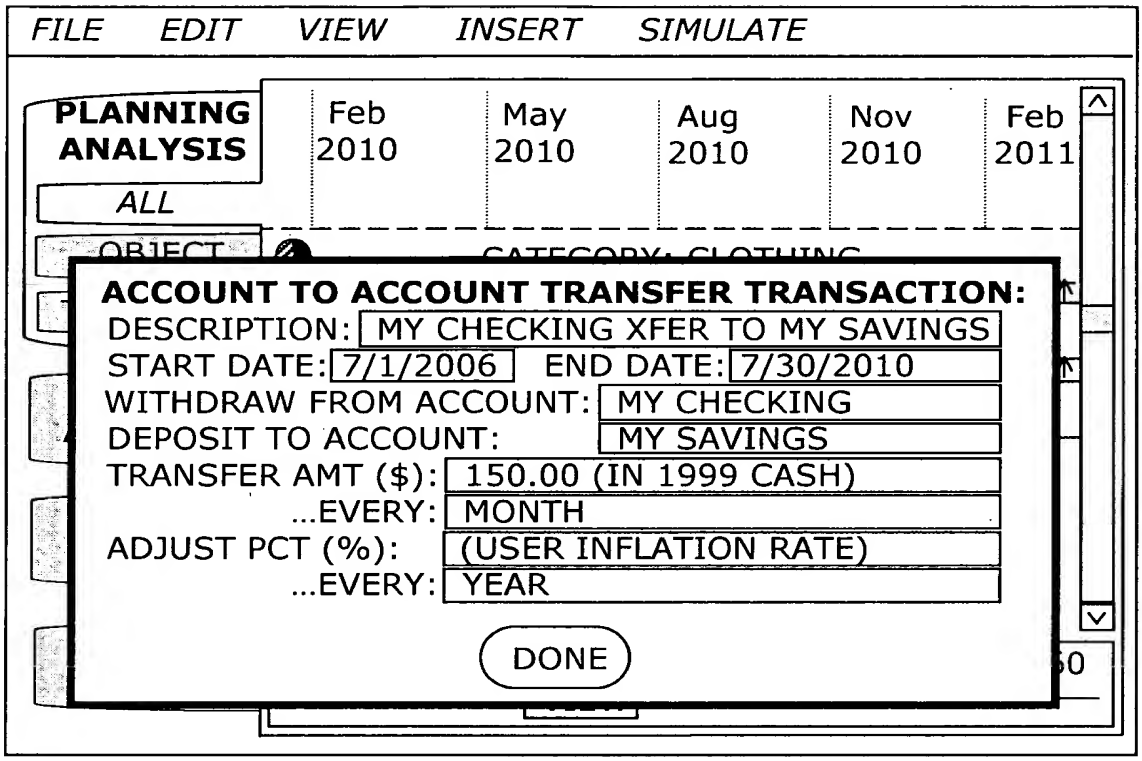


FIG. 41

420

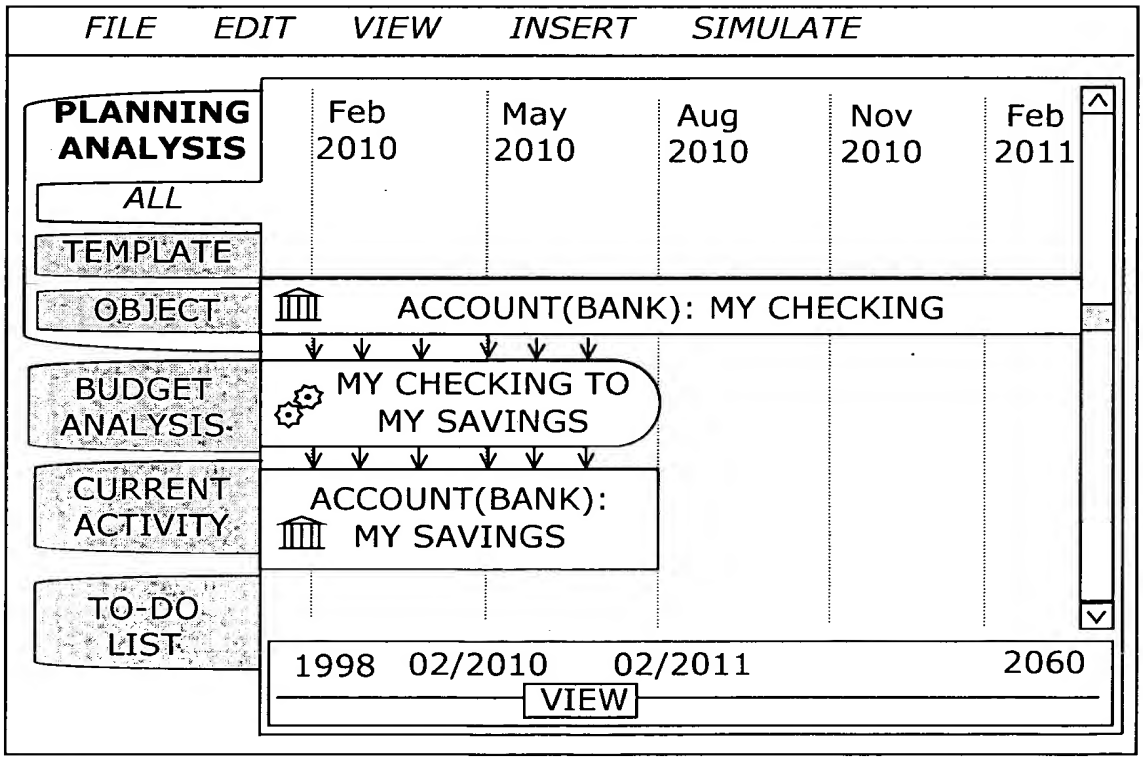


FIG. 42

430

Transaction Class

```
class CTrans : public CtimeValue // A pure-virtual/abstract
                                 //    base class!
{
public:
   CTrans(
      const char *name, // Transaction's reference-name
      ... );                // More input parameters (not shown)
   virtual ~CTrans();

   virtual CDate updateWithDate( CDate date_curr ) = 0;
                 //PURE VIRTUAL!...Must inherit this class!

   virtual CDate resetToFirstDate();
                 // Inherited from Time/Value

   virtual void updateDateRange(
     CDate date_start, CDate date_stop );
                 // Inherited from Time/Value

protected:
   CScheduler  m_schUpdate; // Schedules next update date
   CScheduler  m_schAdjust;  // Schedules next adjust date
   priority_t    m_priority;     // Priority (0=lowest)

}; // END of 'CTrans' class
```

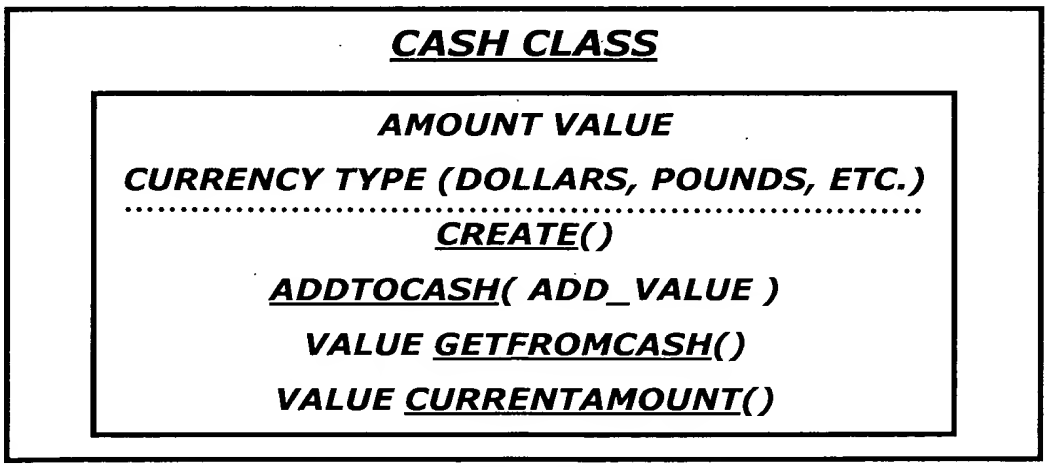
FIG. 43

Account-to-Account Transfer Transaction Class

```
class CTrans_acctToAcct : public CTrans
{
public:
   CTrans_acctToAcct(
      const char *name,  // Transaction's reference-name
      ...     );          // More input parameters (not shown)
   virtual ~CTrans_acctToAcct();

   virtual CDate updateWithDate( CDate date_curr );

   virtual CDate resetToFirstDate();
   virtual void updateDateRange(
      CDate date_start, CDate date_stop );

protected:
   CAccount    *m_acctFrom; // Xfer 'From' this accnt
   CAccount    *m_acctTo;   // Xfer 'To' this accnt
   value_t  m_moneyToXfer;  // Money to transfer at
                            //   each update schedule
   value_t  m_adjustPct;    // % to adjust xfer amount

}; // END of 'CTrans_acctToAcct' class
```

440

FIG. 44



450

FIG. 45

Account-to-Account Transfer
Transaction Class Method Example

```
CDate CTrans_accntToAccnt::updateWithDate( CDate
date_curr ) {
   Cdate date_test =
     SYSINTF.getCurrentDate(); // Not used, just for demo
   if ( date_test == date_curr )
     SYSINTF.print( "Just a test...dates will be equal!" );
   // If simulated current date does NOT match our expected
   //    current date, leave (an invalid condition)
   if ( date_curr != m_schUpdate.getNextDate() ) {
     SYSINTF.throwError( ERR_UNEXPECTED_DATE );
     return(date_curr);// Ends simulation for this transaction!
   }

   // If current simulation date matches or exceeds
   //    our next adjustment date, adjust parameters
   if ( date_curr >= m_schAdjust.getNextDate() ) {
     m_moneyToXfer *= 1.0 + m_adjustPct / 100.0;
     m_schAdjust.computeNextDate(); }  // Set the next
           // adjustment date

   // CREATE 'cash' data type (simulated cash amount = 0)
   CCash cash_xfer;
   // WITHDRAW cash FROM account (simulate cash > 0)
   m_acctFrom->withdraw( m_moneyToXfer, cash_xfer );
   // DEPOSIT cash TO account (simulated cash = 0)
   m_acctTo->deposit( cash_xfer );

   // LOG this transfer amount to the Time/Value (base) class
   addNew( date_curr, m_moneyToXfer );
   // Return the date that we wish the Cash-Flow Simulator to
   //    call us with again
   return( m_schUpdate.computeNextDate() );
   // NOTE: Return from this method will cause 'cash_xfer'
   //    to be destroyed, calling 'cash' class' destructor
   //    method.  A NON-ZERO simulated cash amount in
   //    'cash_xfer' would cause a system warning!
} // END of 'CTrans_accntToAccnt::updatePerDate()'
```

460 →

FIG. 46